Variable Intensity Patches With Swirls (VIPs): Novel Data Augmentation for Retinal Vessel Segmentation

Prateek Jeet Singh Sohi

A Thesis
In The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science (Electrical and Computer Engineering)
Concordia University
Montréal, Québec, Canada

April 2024

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By:	**Prateek Jeet Singh Sohi**
Entitled:	**Variable Intensity Patches With Swirls (VIPs): Novel Data Augmentation for Retinal Vessel Segmentation**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. Nawwaf Kharma ______________________	Chair
Dr. Mahdi S. Hosseini ______________________	External examiner
Dr. Nawwaf Kharma ______________________	Examiner
Dr. Maria Amer ______________________	Supervisor

Approved by: ____

Dr. Zahangir Kabir, Graduate Program Director

______ 2024 ____

Dr. Mourad Debbabi, Dean

Gina Cody School of Engineering and Computer Science

Abstract

Variable Intensity Patches With Swirls (VIPs):

Novel Data Augmentation for Retinal Vessel Segmentation

Prateek Jeet Singh Sohi

A significant problem in retinal vessel segmentation (RVS) research is overfitting mainly due to the lack of large datasets. Data augmentation can alleviate this problem. Current augmentation techniques for RVS do not address the main challenges of localized variable intensities and microvessels in retinal images. This thesis proposes a data augmentation technique (termed variable-intensity patches with swirls, VIPs) to create augmented retinal images by randomly adding variable-size and variable-intensity square patches with swirl structures to a training image, online during training. We add patches and swirls without overlapping the retinal vessels. Our variable patches simulate images with illumination changes, and swirls add microvessel-like structures. To evaluate our augmentation technique, we study recent RVS models and examine the impact of their components, including augmentation, augmentation mode, and preprocessing. We then propose both data preprocessing (gamma correction and contrast enhancement) and VIPs augmentation to address challenges in RVS. Our experiments with in- and cross-datasets show that our combined augmentation and preprocessing technique significantly improves the performance of RVS baseline models (e.g., LWNet by 6.68% and SegRNet by 5.29% in AUC measure). Also, our technique is more stable than all related works across RVS models. Our approach helps to reduce the training-validation losses of RVS models and the gap between training and validation losses. We performed ablation studies on our technique: comparing patches versus swirls, looking at the impact of preprocessing, and analyzing its hyperparameters.

Acknowledgments

I would like to extend my sincere appreciation to my supervisor, Dr. Maria Amer for her invaluable support, unwavering attention, and exceptional attention to detail throughout my academic journey. Her perseverance and guidance played a pivotal role in enabling me to achieve my academic goals.

I would like to express my heartfelt gratitude to my parents for their unwavering and unconditional support throughout my academic journey. Their invaluable presence has been instrumental in my success, and I dedicate this entire thesis to my supervisor and parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Problem Statement

Retinal vessel segmentation (RVS) is essential for early disease detection. State-of-the-art in RVS uses deep networks such as multi U-net [6, 11–13]. A U-net [4] is a fully connected convolutional neural network that utilizes skip connections to facilitate information transfer between it's down-sampler and up-sampler part of the network to improve localization. There are three main challenges with modern RVS approaches:

(a) Datasets of retinal images are tiny (the largest is HRF [3] with 45 images). Small datasets hinder the model's generalization and cause over-fitting. Gathering more data is expensive in the medical domain.

(b) Most RVS models find it difficult to segment microvessels that resemble contours (noise) in the image background [8]. Microvessel segmentation is important for early diagnosis because anomalies in the microvessels are the first signs for the onset of degenerative retinal diseases.

(c) Most retinal images have localized intensity variations that make vessel segmentation difficult [8].

The survey paper [14] presents a comprehensive review of RVS models and reports that only 42% of the papers reviewed use data augmentation for regularization. The survey paper [15] makes two main observations:

(a) Deep learning based RVS techniques struggle to segment microvessels.

(b) RVS techniques do not generalize well when there is domain shift, i.e., applied on different datasets.

Therefore, the main challenges in RVS are:

1. Small size of RVS datasets hinders model generalization [14–16].
2. Localized intensity variations in RVS images make vessel segmentation difficult [8]. (See Figure 1.1 for a visual example.)
3. Microvessel segmentation is difficult as most RVS models confuse them with contours (noise) in the image background [8, 15]. (See Figure 1.1 magnified portion for visual examples.)

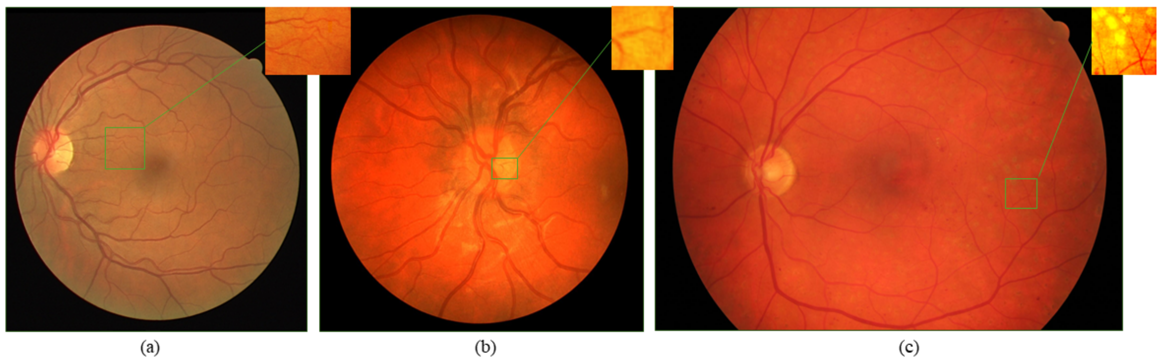

Figure 1.1: Retinal vessel images from different datasets: (a) DRIVE [1] image. (b) CHASEDB [2] image. (c) HRF [3] image.

Data augmentation can address such challenges by increasing the number of training samples and adding patterns to these new samples. The patterns should, however, be appropriate for RVS.

The augmentation methods [9,16] specific to RVS do not solve the above challenges since 1) they do not augment localized variable-intensity patterns, 2) they do not add microvessel like pattern, and 3) they may heavily occlude the vessels as they do not pay attention to their locations. Basic data-augmentation techniques embedded in RVS models are not effective to address these challenges either. They use simple operations such as flipping, rotation, jittering, or blur. Classical data augmentation methods such as random Erase [17], CutMix [10], or Mosaic [18], used in image classification or object detection, have not been applied to RVS. Probably because they may randomly occlude objects (here vessels), resulting in major information loss.

To address the above main challenges in RVS, we propose to use both data preprocessing (offline before training and inference) and data augmentation (online during training at each epoch):

1. Preprocessing, gamma correction or contrast enhancement, helps address localized intensity variations in RV images.

2. Data augmentation that adds both variable intensity patterns and swirls while not occluding the vessels can address all challenges.

1.2 Summary of Contributions

In this thesis, we start by systematically studying the impact of components of RVS models, including the deep learning model itself, the data augmentation it uses, and possible preprocessing operations on training samples it may use. We arrive at the following conclusions.

1. Augmentation is crucial to improve the performance of RVS models. For example, when we evaluated the state-of-the-art RVS model LWNet [6] without its data augmentation, its F1-score dropped by -6.56%. One main reason is the small RVS datasets available for training.

2. RVS models have variable performances across datasets; for example, LWNet [6] performance in the F1-score fluctuates between 80% and 57%.

3. The majority of RVS models use their own data augmentation, that is, adapted to the model architecture. Our simulations show that, indeed, using the augmentation method of one model into another model leads to a major loss in performance.

4. Few augmentation methods exist that are specifically designed for RVS without altering the model's architecture. The challenge is to innovate a method that is stable across RVS models; certainly data augmentation should not introduce performance loss.

5. When adequately selected, data preprocessing can lead to performance gains across RVS models.

To address these conclusions, we first propose the use of preprocessing operations to improve the contrast between the blood vessels and the background and reduces the amount of intensity variations. We perform preprocessing before training and before inferencing. We then contribute a new data augmentation called VIPs (variable intensity patches with swirls). We perform augmentation online during training at each epoch. We add swirls to random-sized square patches in the background (without overlapping the retinal vessels) and randomly vary the intensity of the patches to generate augmented samples. Since swirls resemble unwanted contours that confuse the RVS model, our method provides training examples with more contours than the original training images at random locations in the image background, thus helping the model reduce false negative predictions. By varying the intensity of randomly selected square patches, we increase the number of training images with localized intensity variations to aid model generalization.

1.3 Thesis Outline

Chapter 2 provides a comprehensive analysis of the related work. Section 2.1 provides the necessary preliminary information, Section 2.2 is a review of preprocessing techniques used for RVS. In Section 2.3, we discuss general-purpose augmentation methods related to RVS, while in Section 2.4 a summary of RVS specific augmentation methods is given. Finally, Section 2.5 gives a detailed account of the workings of various RVS models and the data augmentation techniques they use.

Chapter 3 presents a detailed description of our proposed VIPs method and preprocessing operations.

In Chapter 4, Section 4.1 describes our simulation setup, comparison matrices, and datasets used. Section 4.3 quantitatively compares recent RVS baseline models and identifies the best RVS model. Next, in Section 4.4, we study the effect that individual components (that is, data augmentation and preprocessing) have on the best RVS models. Section 4.5 compares our VIPs method with related augmentation methods. This is followed by the ablation studies of our VIPs method in Section 4.6. Section 4.7 then compares the visual results of our proposed VIPs method with other related augmentation methods. Finally, we analyze the effects of VIPs on validation and training loss during model training in Section 4.2.

Chapter 5 summarizes our contributions and provides information about the prospective future work, which concludes this thesis.

Chapter 2

Related Works

Our thesis proposes to use both preprocessing and data augmentation to improve performance of RVS models. Thus, this Chapter reviews related work to preprocessing (Section 2.2), general-purpose data augmentation (Section 2.3), augmentation methods specific to RVS (Section 2.4), and RVS models and their data augmentation (Section 2.5).

2.1 Preliminaries

Data augmentation aims to increase the number of training samples by generating modified copies of the original data. The modifications are achieved through simple image transformations such as scale or blur or through sophisticated image operations such as adding whole objects. Depending on the task and objectives, the modification may lead to minor or major changes. The increase in training samples aims at better generalization of deep learning models.

Data augmentation may be performed either offline (the dataset is expanded with augmented samples before training) or online (the training images are augmented at each training epoch). Online augmentation has the advantage of producing different augmented images at each epoch; the model gets thus, trained on a larger number of unique images compared to offline augmentation. The probability p of using a specific transformation for augmentation in online modes is referred to as the p-value. While in offline mode p-value refers to the percentage p of training images used for augmentation out of all training images.

Preprocessing, on the other hand, enhances the image dataset before training, using a set of transformations such as contrast enhancement. A critical difference between offline data augmentation and preprocessing is that the number of training images remains unchanged after preprocessing, that is, all images are preprocessed.

2.2 Data Preprocessing and RVS

There is little research on preprocessing methods for RVS. The RVS model AR-SA-U-net [19] performs data preprocessing by extracting the green channel, followed by CLAHE [5] (that is, contrast limited adaptive histogram equalization). The authors do not mention the clip limit and gamma values. The RVS model Wave-Net [20] performs preprocessing by converting all images to grayscale, then applying CLAHE [5] followed by gamma correction with γ =1.2, but the clip limit used for CLAHE [5] is not mentioned by the authors. The model Res2Unet [21] performs data preprocessing by extracting the green channel, followed by CLAHE [5] and gamma correction. The authors do not mention the clip limit and gamma values. To our knowledge, SegRNet [7] is the only RVS paper that provides a complete set of hyper-parameter values used for preprocessing for RVS. SegRNet [7] performs preprocessing by converting all images to grayscale, then applying CLAHE [5] with a clip limit of 3 and grid size of 8×8, followed by gamma correction with γ =1.2.

The paper LIOT [22] proposes a preprocessing method for segmenting curvilinear objects (including cracks and retinal vessels). LIOT [22] compares the pixel values of the input image $I(p)$ with a group of 8 neighboring pixels lying in the perpendicular direction. The comparison made with groups on the top, bottom, left, and right side for the individual channels form a 4-channel output image $I(p_s)$ given by Equation 1.

$$\begin{aligned} &I(p_s) = \sum_{1}^{8}[I(p) > I(p_s^i)] \times 2^{i-1}, \\ &[I(p) > I(p_s^i)] = 1, \\ &[I(p) < I(p_s^i)] = 0. \end{aligned} \tag{1}$$

Where, $I(p_s^i)$ is the pixel value at a Euclidean distance i from $I(p)$ towards a direction s (that is either towards the top, bottom, left, or right).

2.3 General Purpose Data Augmentation

This section reviews general-purpose augmentation methods, meaning those used across traditional computer vision domains such as image classification and object detection.

The patch-based random Erase [17] randomly occludes image patches by setting pixel values inside the randomly selected patches to zero, which makes the classification models more robust to occlusion. CutMix [10] augments an image by occluding it with patches belonging to another image to get the train set. Mosaic [18,23] combines random portions of four images to form the augmented sample. To the best of our knowledge, such sophisticated augmentation methods have not been used in RVS. One reason could be that these patch-based methods do not pay attention to the object's location (here, vessel) and may randomly occlude them, resulting in information loss.

2.4 Augmentation Methods Specific to RVS

There are papers such as [9,16] proposing data augmentation methods specific to RVS without altering the model itself. The paper [16] shows that using transfer learning (that is, they use pre-trained weights for a selected number of layers and train the remaining layers on the target dataset) in addition to their proposed data augmentation (random rotation, cropping, translation, shear, zoom, CLAHE, saturation, sharpening, mean-shift blur, and gamma correction in the offline mode) improves U-net [4] model performance. The authors of [16] do not mention the p-values used for the augmentation.

The online data augmentation method [9] for RVS first performs vertical and horizontal flipping with p=0.5 followed by channel-wise random gamma correction, which takes the input image V_i and outputs an RGB image V_i' with one of its channels transformed using gamma correction as given in Equation 2.

$$V_i'[:,:,k] = V_i[:,:,k]^{\gamma_i}, \qquad \gamma_i \in [0.33, 3]. \tag{2}$$

Where, V_i' is the output image with an image channel transformed using gamma correction, k is the randomly selected image channel in the set $\{R, G, B\}$ corresponds to the index values of the red, green, and blue color channels of the input image V_i, γ_i is the gamma value used for gamma correction. Finally, channel-wise random vessel augmentation applies a morphological transformation on the transformed image V_i' that alters vessel geometry by omitting vessels less than a certain thickness and generates a rough segmentation map M_i. The final augmented image is given by V_i^t as shown in Equation 3.

$$V_i^t = V_i'(1 - M_i) \times \beta + M_i \times \beta, \qquad \beta \in [0, 1]. \tag{3}$$

Where, V_i^t represents the final augmented image, M_i is a rough segmentation map, V_i' is the transformed image obtained after random channel-wise gamma correction, and β is a control parameter.

Note that augmentation methods for RVS such as [9, 16] use gamma correction as part of their augmentation strategy, while RVS models such as [7, 19–21] use gamma correction for preprocessing.

As [9, 16], our method is also specific to RVS, that is, we propose a data augmentation to improve the model's performance without altering the RVS model itself. The main differences between our VIPs and [9, 16] are as follows:

(a) They do not use any preprocessing.

(b) We utilize ground truth images to maintain the original structure of retinal vessels while altering only the background.

(c) The swirl component of our method is specifically designed to improve the segmentation of microvessels.

(d) Methods [9, 16] do not attempt to solve the issues caused by localized intensity variation, while the patch component of our VIPs method is specially designed to address this problem.

(e) Unlike [9, 16] that only test there method on a single model, we evaluate our method across RVS models and show it is stable (useful) .

2.5 RVS Models and their Data Augmentation

In this section, we categorize and present recent RVS models and their data augmentation methods.

Many RVS models such as [6, 8, 11–13] are based on the U-net [4] architecture, which is a fully connected convolutional neural network architecture that utilizes skip connections to facilitate information transfer between it's down-sampler and up-sampler part of the network for improving localization. A skip connection passes the output of a layer as input to a layer by skipping a few layers in between.

Many U-net based RVS models have multi U-Net architecture. In Figure 2.1, we present a generalized representation of multi-U-net architectures. The main idea behind the design of any multi-U-net architecture is to improve the model's performance by cascading the output of one U-net [4] to the next mini-U-net, which could lead to better feature refinement, and an overall better model performance.

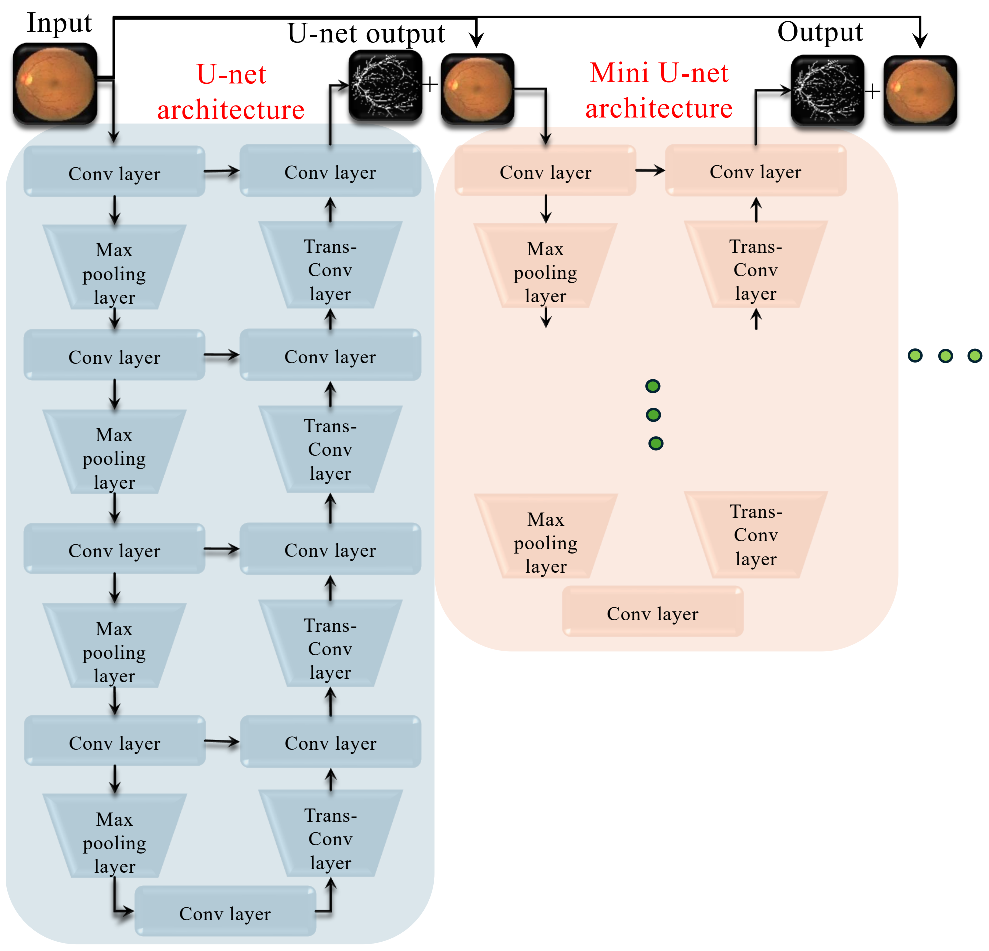

Figure 2.1: Typical multi-U-net network widely used in RVS. A U-net [4] is a full connected convolutional neural network used for medical image segmentation. A mini-U-net has fewer down-sampling and up-sampling layers. The depth and number of the mini-U-net varies depending on the RVS model. The output of a mini-U-net is cascaded as input to the next mini-U-net.

2.5.1 Multi-U-net Networks

The RVS model IterNet [8] passes the output of the ultimate layer of a U-net [4] as input to a series of 2 mini-U-nets connected in series and concatenates the output of each U-net [4] to form the final segmentation map. For augmentation, IterNet [8] uses random scale, shear, and shift with probability $p = 0.05$; random contrast, saturation, brightness, rotation, horizontal and vertical flipping operations with $p = 0.5$ in the online fashion. The models CRAUNet [13] and LWNet [6] use two U-nets [4] connected in series. To improve information transfer between the down-sampler and up-sampler part of the network, CRAUNet [13] uses a self-attention block called MFCA and DropBlock [24] regularization to reduce over-fitting. For augmentation, vertical and horizontal flipping and random rotation are used in an online fashion (the authors do not mention the p values). For augmentation, LWNet [6] performs random scale, translate, and rotate with p=0.33, color jitter with p=1, and horizontal and vertical flipping with p=0.5 in the online mode. FR-Unet [12] uses full-resolution convolution layers instead of skip connections (typically used in U-net [4]) for learning spatial features between the network's down-sampler and up-sampler side. The model FR-Unet [12] performs online augmentation using horizontal and vertical flipping with $p = 0.5$ and random rotation with $p = 0.75$. DCU-Net [11] cascades two U-nets in series and uses a channel attention module to transfer information from the first U-net's [4] up-sampler side to the second U-net's [4] corresponding down-sampler side. It uses deformable convolution blocks [25] instead of convolutional blocks that improve the model's ability to capture vessel geometry. DCU-Net [11] crops the images to 48 $\times$ 48 and performs vertical and horizontal flipping to achieve augmentation. DCU-Net [11] performs preprocessing by extracting the green channel of the image and applying CLAHE [5] to it (the authors do not report the clip limit used for CLAHE [5]).

2.5.2 Multi-branch Networks

CSGNet [26] is a fully connected network that consists of three branches, each branch is trained on images scaled by 0.25, 0.5, and 1. The output of the lower-resolution branches is passed as input to the higher-resolution branch, and all outputs are concatenated to form the final output image. For augmentation, CSGNet [26] performs random rotation, mirror, and scaling, but the authors do not mention other details, such as the mode or p-values. ILU-Net [27] proposes two down-sampling and up-sampling blocks for U-net [4], namely DIB and UIB, which are symmetric blocks consisting of three branches, where the first branch consists of a 1×1 convolutional layers with the second and third branches consisting of an additional one and two 3×3 convolutional layers respectively. The output of the DIB and UIB blocks is a concatenation of the feature map produced by their three branches. For augmentation, ILU-Net [27] performs random rotation, offset, shear, scaling, horizontal flipping, and SAMSIE [28] (spatially adaptive multi-scale image enhancement) in the offline mode. However, the authors of ILU-Net [27] do not mention the p-values. AT-CNN [29] is a U-net [4] inspired architecture with two branches on the down-sampler side, where the first is a convolutional neural network (CNN), and the second one is an axial transformer. The output of the two branches is then fused and passed on to the up-sampler side. Using axial self-attention with CNN for encoding helps capture local and global features. For augmentation, AT-CNN [29] performs random flipping on cropped input images of size 48×48 in the offline mode, and the p-values are not mentioned by the authors. GDF-Net [30] consists of two mini-U-nets with their proposed ASPP module instead of skip connections, the ASSP module passes the input feature map parallelly to 4 dilated convolutional layers with dilation rates of 1, 6, 12, and 18 and concatenates their output. The two U-nets [4] individually focus on global features and the enhancement of local features. The outputs are fused with the input image and passed to an attention-based fusion network that is also a modified U-net [4].

For augmentation, GDF-Net [30] performs random cropping, but the authors do not mention other details, such as the mode or p-values. DPF-Net [31] is a two-path U-net [4] like architecture where the first branch is a dense convolutional neural network and the second branch is a recurrent convolutional neural network. It fuses the multi-scale feature maps from each branch and passes them to the up-sampler side using its interactive fusion block. DPF-Net [31] performs random cropping to obtain patches of size 64×64, which are used as augmented samples.

In conclusion, multi-branch network architectures claim to have superior segmentation capabilities for small and larger blood vessels by utilizing different-sized convolution kernels for different branches, which incorporate both global and local features.

2.5.3 Networks Using Dilated Convolutions

The RVS model SegRNet [7] captures retinal blood vessel morphology using a DFM block that magnifies the vessels and obtains denser features by performing dilated convolutions using a DMFF block. For augmentation, SegRNet [7] performs random brightness, motion blur, grid distortion, optical distortion, rotation, vertical flip, horizontal flip, elastic transform, median blur, and random brightness contrast transformations in the offline mode with p=1. It has a preprocessing step that converts all images to grayscale, then applies CLAHE [5] with a clip limit of 3 and grid size of 8X8 followed by gamma correction with γ =1.2. The RVS model AR-SA-U-net [19] is a U-net [4] variant that uses dilated convolutions to solve the issue of vanishing gradients in the residual module. It proposes a new SCSE [32] based attention module added as an intermediate block during downsampling and upsampling. AR-SA-U-net [19] performs augmentation offline by cropping all the input images using a fixed-size sliding window with p=1. It has a data preprocessing step by extracting the green channel, followed by CLAHE [5]. The authors do not mention the clip limit and gamma values. The model MAGF-Net [33] proposes two new network blocks, MSA and HFP. The MSA specializes in multi-scale feature extraction by cascading three 3×3 consecutive convolutional layers to obtain respective fields of 5×5 and 7×7. The HFP block solves the problem of information losses caused by max-pooling during downsampling. HPF downsamples the dot product of Maxpooling, Avgpooling, and 3×3 convolutional operations performed on the feature maps. GT-DLA-dsHFF [34] is a U-net [4] variant that adds a transformer between the ultimate layer of the down-sampler and the input layer of the up-sampler to extract long-distance features. A dual local attention module was introduced that used dilated convolutions to increase the receptive field and unsupervised edge detection to preserve edge information to aid with information transfer between corresponding down-sampler and up-sampler blocks. It augments the data by performing random rotation, horizontal flip, and color enhancement, but the authors do not mention other details, such as the mode or p-values.

The main difference between the methods discussed in this section and others is their use of dilated convolutions as a means of increasing the receptive field, which leads to better extraction of global features, which could be a significant contributor to their overall performance. We provide a pictorial representation of standard and dilated convolution blocks in Figure 2.2.

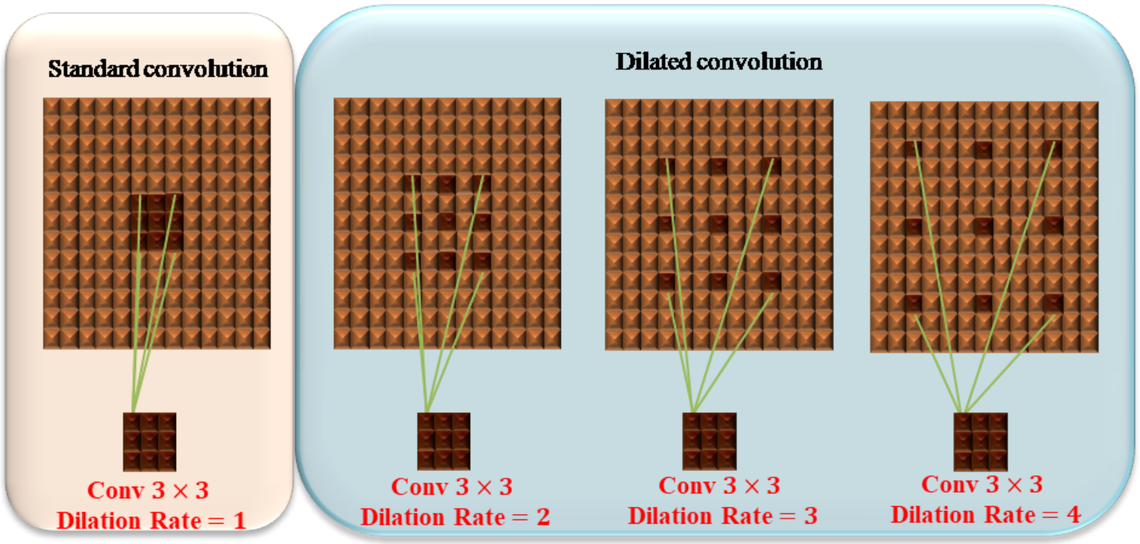

Figure 2.2: Visual representation of standard and dilated convolution with different dilation rates. Dilated convolutions are used for increasing the receptive field of a layer without incurring losses in resolution.

2.5.4 Networks with Novel Modules

These are the networks that do not fall into other categories but propose new models of their own. A recent U-net [4] modification ResDO-UNet [35] proposes a convolution layer called the DO-Conv layer that can extract robust context features, thus increasing segmentation performance. ResDO-UNet [35] performs random cropping for augmentation. The model WANet [36] incorporates a layer of second-order differential Gaussian filter (DSD-GMF) that provides feature maps associated with specific vessel width, which are passed to a width attention module that captures channel and position co-relation between the DSD-GMF feature maps and assigns them with appropriate weight. For augmentation, WANet [36] uses random Gaussian blur, flipping, rotation, cropping, and border crop in the online fashion. Wave-Net [20] is a U-net [4] based architecture that uses a detail enhancement and de-noising block (DED) instead of skip connections. A DED block consists of a 5 × 5 convolutional layer followed by a 1 × 1 convolutional layer designed to capture spacial information and a denoiser consisting of a 3 $\times$ 3 convolutional layer concatenated with a 3 $\times$ 3 raw feature map followed by a 1 $\times$ 1 convolutional layer. For augmentation, Wave-Net [20] performs random cropping, but the authors do not mention other details, such as the mode or p-values. Wave-Net [20] performs preprocessing by converting all images to grayscale, then applying CLAHE [5] followed by gamma correction with γ =1.2; the clip limit used for CLAHE [5] is not mentioned by the authors. The model VG-DropDNet [37] uses VGG blocks on the down-sampler and up-sampler side of a U-net [4] shaped architecture and replaces the skip connections with a DenseNet [38] consisting of a dropout layer after each layer to avoid over-fitting. VG-DropDNet [37] performs augmentation using random cropping with a fixed window size of 64 $\times$ 64 in the offline mode with p=1. The model PLVS-Net [39] is a U-net [4] like architecture that captures spatial features with a prompt block on the down-sampler side. The prompt block first generates an asymmetric feature map by passing the input feature map through a 1 $\times$ 3 and 3 $\times$ 1 convolutional layer and concatenating them; this asymmetric feature map is then concatenated with two more feature maps, each obtained by passing the input feature map through a 3 $\times$ 3 convolutional layer and transposed convolutional layer followed by batch normalization and ReLu layer. For augmentation, PLVS-Net [39] performs brightness, color, contrast enhancement, random flipping, and random rotation. The RVS model BFMD-SN-U-net [40] adds a switchable normalization [41] (SN) layer at the end of every convolutional layer in addition to using a BFMD module to reduce overfitting and improve generalization. They use an attention module called GCI-CBAM to aid information transfer from the down-sampler to the up-sampler side of the network to improve generalization. BFMD-SN-U-net [40] performs data augmentation as mentioned in [16]; that is, they perform random rotation, cropping, translation, shear, zoom, CLAHE, saturation, sharpening, mean-shift blur, and gamma correction in the offline mode. However, the authors of BFMD-SN-U-net [40] and [16] do not mention the p-values they use. The model MFI-Net [42] is a U-net [4] like architecture that performs parallel attention learning using a PSE module, which is a multi-scale extension of the SE [43] module that is used to capture the variable widths of retinal vessels. To effectively capture the inconsistent semantic features, MFI-Net [42] proposed a C2F module that transfers information between the down-sampler and up-sampler sides. For augmentation, MFI-Net [42] employed random flipping and rotation in the offline mode, the p-values are not mentioned by the authors. BST-DSN model [44] fuses the output obtained from the ultimate layer of each VGG block on the up-sampler side to obtain the segmented image. For augmentation, BST-DSN [44] uses rotation, flipping, and scaling operations to perform offline. Deep-Ret [45] proposes a customized residual network that utilizes a novel loss function containing an orientation diversity term that ensures desirable orientation sensitivity and a noise robustness term to minimize the effects of noise. The recent model Res2Unet [21] integrates the backbone Res2Net [46] into U-net [4] in addition to channel attention for providing more robust predictions. Res2Unet [21] performs the data preprocessing step by extracting the green channel, followed by CLAHE and gamma correction. The authors do not mention the clip limit and gamma values. Res2Unet [21] does not do data augmentation. The model CS-net [47] uses channel and spatial attention modules to improve vessel localization by increasing information gain. It performs vertical flipping with $p = 0.5$ and random rotation with $p = 1$ in the online fashion. It uses a channel attention block to effectively utilize multi-channel spaces for feature representation. D-GaussianNet [48] uses Gaussian-matched filters to estimate vessel curvature for the enhancement of spatial attention. The authors of D-GaussianNet [48] do not provide details about the augmentation method used.

The segmentation methods described in this section do not have a common underlying concept to explain their overall performance. Instead, they rely on uniquely designed modules to achieve better segmentation.

2.5.5 GAN Based Network

SEGAN [49] is a U-net [4] inspired generative adversarial network (GAN) that performs flipping and rotation operations to achieve data augmentation; the authors do not mention the exact hyper-parameter values and mode of augmentation.

2.5.6 GNN Based Network

Inspired by graph neural networks (GNN), BTU-Net [50] replaces the U-net [4] block with a unidirectional graph that is optimized using the BTLBO [51] algorithm to find the best block structure. It follows the data augmentation regime of ILU-Net [27].

2.5.7 Cross-domain Network

Curvilinear objects include cracks as well as retinal vessels. Models for curvilinear object segmentation (COS) include JTFN [52] and LIOT [22]. COS papers claim to be effective for both retinal and crack image segmentation. However, our simulation shows that COS models such as JTFN [52] cannot compete with retinal-specific models such as LWNet [6] in RVS. For example, the F-score under cross-dataset CHASE-DB to DRIVE is 68.76% versus 80.18%. (Metrics and datasets are explained in the simulation Section 4.1.) JTFN [52] is a U-Net-based model that performs feature refinement using a feature interactive module and a gated attention unit. It has two branches on the up-sampler side, one for boundary detection and another for boundary refinement. The feature interactive module facilitates information transfer between the boundary detector and the segmentation branch. For augmentation, JTFN [52] utilizes color jitter, rotate, horizontal, and vertical flip transformations with $p = 0.5$.

Chapter 3

Proposed Approach

Studying RVS models and their components, we conclude that few models use preprocessing and that augmentation is often specific to the model architecture itself. Also, we observe that related augmentation methods for RVS do not address the problem of localized variable intensities in retinal vessel images, nor do they augment patterns to simulate background contours that the models confuse with microvessels. To address these limitations, we propose to use both preprocessing, to reduce localized intensity variations in retinal vessel images, and data augmentation to add (augment with) localized variations and microvessels.

We use preprocessing at training and inference time; that is, all images of a dataset are preprocessed (enhanced) before RVS. Our experiments in Section 4.4 show that the best preprocessing operations for RVS are contrast enhancement (specifically the contrast-limited adaptive histogram equalization CLAHE from [5]) followed by gamma correction on the grayscale image (as proposed in SegRNet [7]). The CLAHE [5] transformation step of the pre-processing reduces localized intensity variations in the image, while the gamma correction step improves the contrast between the image background and retinal vessels. We provide visual examples obtained after each step of the preprocessing pipeline using an original DRIVE [1] image in Figure 3.1.

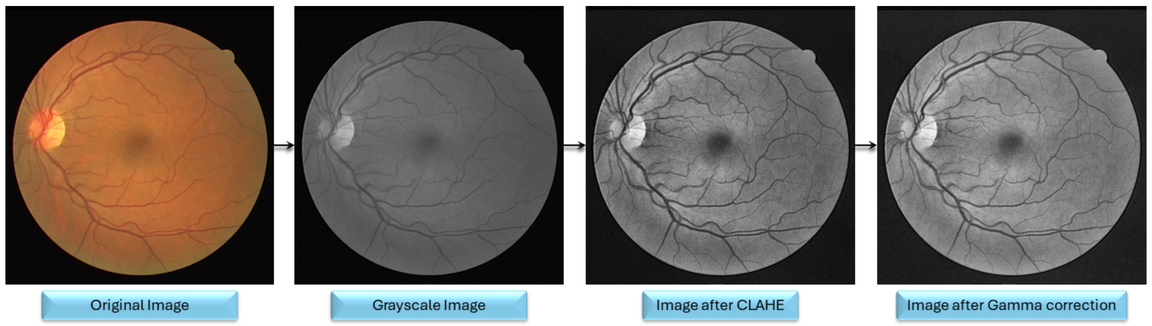

Figure 3.1: Visual examples depicting the outputs of the various steps used for preprocessing(i.e., grayscale conversion followed by CLAHE [5] and gamma correction) performed on an original DRIVE [1] image.

We propose a data-augmentation technique specific to RVS without altering the architecture of the RVS model. We perform data augmentation online during training; that is, at each training epoch, the augmentation is applied to an image with a probability p. Augmentation at inference decreases inference speed and is, hence, typically avoided. Our variable-intensity patches with swirls (VIPs) method creates augmented retinal images by randomly adding variable-size and variable-intensity square patches with swirl structures to a training image in online mode. We add patches and swirls without overlapping the retinal vessels. Our variable patches simulate image variations such as illumination changes, and our swirls add vessel-like structures.

We provide visual samples of original images from DRIVE [1], CHASEDB [2], and HRF [3] versus their VIPs counterparts in Figure 3.2.

The flow chart in Figure 3.3 allows an appreciation of the steps of the proposed VIPs listed in Algorithm 1, which takes the input image img and its corresponding binary map gt and produces an augmented image aug. As seen, the hyper-parameters are α, β, and γ. The total number of patches to add equals $\alpha\times$ the width of the input image. Our intensity transformation multiplies pixels of each patch by a uniformly random number selected from the interval $[\beta, 1]$. We induce swirls using a strength threshold randomly chosen from $[0, \gamma]$.

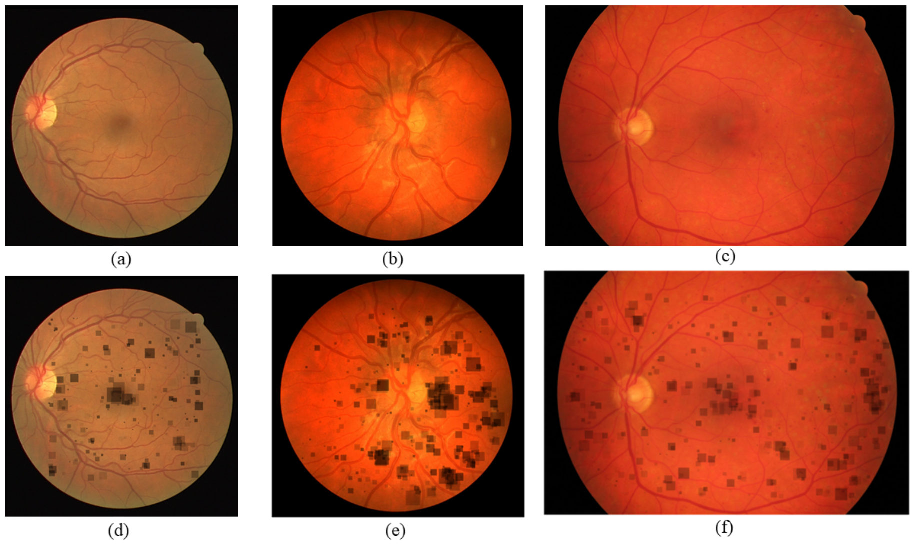

Figure 3.2: (a) Original DRIVE [1] image. (b) Original CHASEDB [2] image. (c) Original HRF [3] image. (d) Original image in (a) augmented with our VIPs. (e) Original image in (b) augmented with our VIPs. (f) Original image in (c) augmented with our VIPs.

Our algorithm starts by randomly selecting variable-sized square patches p_{img} of width p_{size}, with (p_x,p_y) being the coordinates of the anchor points over img. The patch is an image window, $img[px : p_{size} + px, py : p_{size} + py]$, which stretches from p_x to p_x+p_{size} in the x-direction and from p_y to p_y+p_{size} in the y-direction. A retinal image consists of vessels in the camera's field of view and a black background outside the field of view. We ensure that the randomly selected patches do not overlap with the retinal vessels by discarding the patches that contain non-zero pixels in their ground truth. Patches with even one zero-valued pixel are also discarded to ensure the patches are inside the field of view, that is, $max(p_{gt}) = 0$ and $min(p_{img}) \neq 0$. Then, we apply intensity variation and add a swirl to the selected patch. Our intensity transformation for the selected patch can be summarized using Equation 4.

$$p_{img} = img[px : p_{size} + px, py : p_{size} + py] \times \omega, \qquad \omega \in [\beta, 1]. \tag{4}$$

Where, p_{img} is the image patch after the application of our intensity transformation, $img[px : p_{size}+px, py : p_{size}+py]$ is the selected image patch, ω is the intensity control

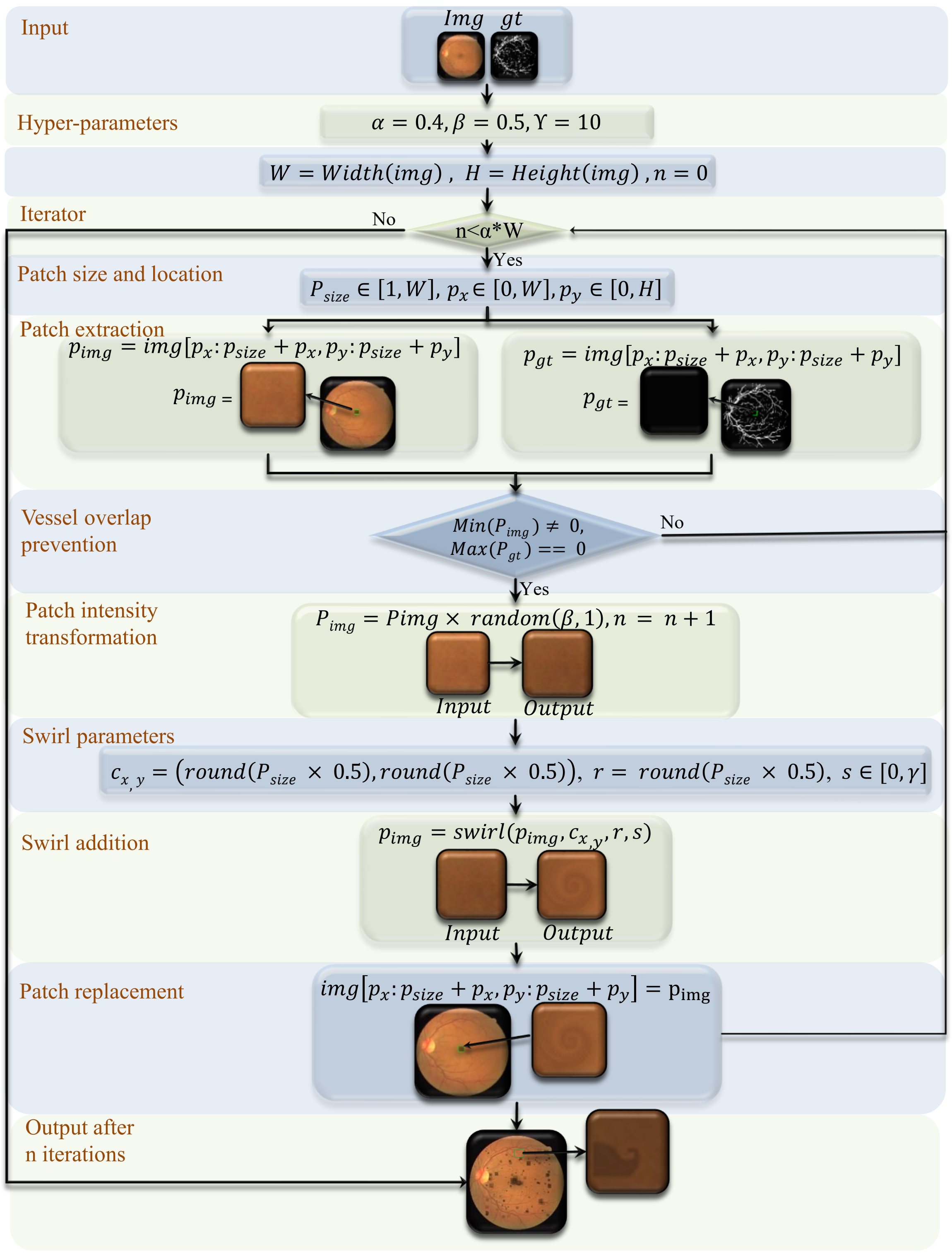
Input
Img gt
Hyper-parameters
α = 0.4, β = 0.5, ϒ = 10
W = Width(img) , H = Height(img) , n = 0
Iterator
No
n<α*W
Yes
Patch size and location
P_size ∈ [1, W], p_x ∈ [0, W], p_y ∈ [0, H]
Patch extraction
p_img = img[p_x: p_size + p_x, p_y: p_size + p_y]
p_img =
p_gt = img[p_x: p_size + p_x, p_y: p_size + p_y]
p_gt =
Vessel overlap prevention
Min(P_img) ≠ 0, Max(P_gt) == 0
No
Yes
Patch intensity transformation
P_img = Pimg × random(β, 1), n = n + 1
Input
Output
Swirl parameters
c_x,y = (round(P_size × 0.5), round(P_size × 0.5)), r = round(P_size × 0.5), s ∈ [0, γ]
Swirl addition
p_img = swirl(p_img, c_x,y, r, s)
Input
Output
Patch replacement
img[p_x: p_size + p_x, p_y: p_size + p_y] = p_img
Output after n iterations

Figure 3.3: Flowchart of our proposed VIPs method depicting the process of patch selection, intensity, and swirl transformation used for artificial simulation of microvessel.

parameter, and β is the minimum possible value of the intensity control parameter.

We iteratively perform the patch selection and augmentation process until we have the desired number of patches based on α; that is, we add $\alpha \times W$ patches.

Algorithm 1 Proposed VIPs

```
1: **Input** $img, gt$
2: **Output** $aug$
3: **Define:**
   patch quantity $\alpha$=0.4
   intensity control $\beta = 0.5$
   swirl control $\gamma = 10$
   number of iterations $n = 0$
4: $W \leftarrow$ width of img
5: $H \leftarrow$ height of img
6: **while** n $< \alpha \times W$ **do**:
7:     Randomly select patch size:
       $p_{size} \in [1, W]$
8:     Randomly select anchor point:
       $(px \in [0,W], py \in [0, H])$
9:     Select patch:
       $p_{img} \leftarrow img[px : p_{size} + px, py : p_{size} + py]$
10:    Select gt patch:
       $p_{gt} \leftarrow gt[px : p_{size} + px, py : p_{size} + py]$
11:    **if** $max(p_{gt}) = 0$ and $min(p_{img}) \neq 0$ **then**
12:        $n \leftarrow n + 1$
13:        Perform intensity transformation:
           $p_{img} \leftarrow p_{img} \times random(\beta, 1)$
14:        Define swirl center:
           $c_{x,y} \leftarrow (round(p_{size} \times 0.5), round(p_{size} \times 0.5))$
15:        Define swirl radius: $r \leftarrow round(p_{size} \times 0.5)$
16:        Define swirl strength: $s \leftarrow random(0, \gamma)$
17:        Perform swirl transformation:
           $p_{img} \leftarrow swirl(p_{img}, c_{x,y}, r, s)$
18:        $img[px : p_{size} + px, py : p_{size} + py] \leftarrow p_{img}$
19:    **end if**
20: **end while**
21: $aug \leftarrow img$
```

The process of swirl generation is described in Algorithm 2 (based on a code of the standard scikit-image library [53] of Python). We use each pixel's Cartesian coordinates (x, y) in the selected patch to obtain its polar coordinates (η, θ) and transform the angle θ using the parameters r (swirl radius) and s (swirl strength).

The transformed polar coordinates (η,θ') are given by Equation 5.

$$\begin{aligned}
&(\eta, \theta') = (\eta, s \times \exp^{\frac{-\eta}{\eta'}} + \theta), \\
&\eta' = \frac{ln(2) \times r}{5}, \\
&(\eta, \theta) \leftarrow (\sqrt[2]{(y - c_y)^2 + (x - c_x)^2}, \arctan(\frac{y - c_y}{x - c_x})).
\end{aligned} \tag{5}$$

Where, (η, θ') represent the transformed polar coordinates, r corresponds to the radius of the added swirl, (c_x, c_y) denotes the Cartesian coordinates of the center point of the added swirl, s is the strength of the added swirl, and (x, y) are the Cartesian coordinates within the selected patch. Finally, we convert the transformed polar coordinates to Cartesian and replace the pixel values at (x, y) with those at the transformed coordinates (x', y') to obtain a swirl on the image patch. The process of adding swirls can be summarized using Equation 6.

$$\begin{aligned}
&swirl_{patch} = p_{img}[x', y'], \\
&(x', y') = (\eta \times \cos(\theta'), \eta \times \sin(\theta')).
\end{aligned} \tag{6}$$

Where, $swirl_{patch}$ represents our selected patch after swirl addition, (η, θ') are the transformed polar coordinates obtained from Equation 5, (x', y') are the transformed Cartesian coordinates used for swirl addition.

Algorithm 2 *swirl* generation

```
1: Input: image patch $p_{img}$, swirl center $(c_x, c_y)$, strength $s$, radius $r$
2: Output: $swirl_{patch}$
3: Initialize: iteration parameters $x = 0, y = 0$
4: $swirl_{patch} \leftarrow p_{img}$
5: for $x \leq p_{size}$ :
6:   for $y \leq p_{size}$
7:     Convert (x,y) to polar co-ordinates:
       $(\eta, \theta) \leftarrow (\sqrt[2]{(y - c_y)^2 + (x - c_x)^2}, \arctan(\frac{y-c_y}{x-c_x}))$
8:     Transform polar co-ordinates to add a Swirl:
       $(\eta, \theta') \leftarrow (\eta,\ s\times \exp^{\frac{-\eta}{\eta'}} + \theta)$ ; where $\eta' = \frac{ln(2)\times r}{5}$
9:     Convert transformed polar to cartesian co-ordinates:
       $(x', y') \leftarrow (\eta \times \cos(\theta'), \eta \times \sin(\theta'))$
10:    Replace pixel values in the patch with pixel values
       for the transformed co-ordinates:
       $swirl_{patch}[x, y] \leftarrow p_{img}[x', y']$
11:  end
12: end
```

The hyper-parameters of the proposed method are α to control the number of patches, β as minimum intensity level, γ as maximum swirl strength, and the p-value, the augmentation probability at each epoch. A γ=0 means only patches are added to the image, and a β=1 means only swirls are added. We select the hyper-parameters values to p=0.5 (that is, the probability of an image being augmented by our method at each epoch, one reason for choosing p=0.5 is that it is the most commonly used p-value in the literature), α=0.4 (that is, the total number of patches to be added to the image are 0.4 times the width of the original image, we observed that increasing the number of patches resulted in a high amount of overlapping between the selected patches which results in 0 valued regions of occlusion, while using a lesser number of patches is insufficient to improve the models robustness to localized intensity variations), β=0.5 (meaning, the intensity level inside the selected patch lies in between 0.5 to 1 times the original image, we visually observed that setting β to a lower values cause a lot of pixel values to become close to zeros which effectively causes occlusion in almost all the patches, while setting it to higher values leads to insufficient intensity variations), and γ=10 (meaning that the maximum swirl strength inside the patch can be 10, we selected this value to ensure that our swirls closely resemble the contours of the image background, at higher value of γ the swirls become too prominent and diss-similar for the original background while lower value result in negligible swirl component). See also Section 4.6 for a thorough study of the hyper-parameters.

Chapter 4

Experimental Analysis

In this Chapter, Section 4.1 explains the simulation setup. Section 4.3 determines which are the best performing RVS baseline models. Section 4.4 demonstrates the impact of data augmentation operations, its mode, and preprocessing on RVS to examine the best option. In Section 4.5, we apply our VIPs and related data augmentation methods on top of the best and second best baselines to perform a comparative analysis. In Section 4.6, we produce an ablation study of our VIPs method. Section 4.2 shows the effect of our VIPs method during model training using training versus validation loss curves. Finally, in Section 4.7, we provide visual results.

4.1 Simulation Setup

The baseline RVS models we used for the experiments were: WANet [36] (2022), FR-Unet [12] (2022), IterNet [8] (2020), SegRNet [7] (2023), and LWNet [6] (2022); we trained them with the same hyper-parameters and the same data augmentation as mentioned by the authors.

We selected three widely used RVS datasets in our simulations:

(a) DRIVE [1] consists of 40 images (565×584), 20 images for training, and 20 for testing.

(b) CHASEDB [2] consists of 28 images (700×605), 20 images for training, and 8 for testing.

(c) HRF [3] consists of 45 high-resolution images (3504×2336), 15 images for training, and 30 for testing.

Most RVS datasets do not define validation sets but only train and test sets. Thus, RVS papers [6, 22, 26, 54, 55] use two types of dataset analysis in-dataset and cross-dataset.

(a) **In-dataset analysis:** the RVS models are trained and tested on the train and test split of the same datasets. This can be considered the validation step in the train-validate-test simulation framework.

(b) **Cross-dataset analysis:** the models are trained on the train set of one dataset but tested on the test set of another dataset. Cross-dataset analysis can be seen as the test phase in the context of the train-validate-test simulation framework.

Since RVS datasets are small, cross-dataset analysis is essential to ascertain the generalization capability of RVS models.

Performance measures (metrics) used for RVS are Precision (P), Recall (R), F1 score (F), the area under the precision recall curve (AUC), and Accuracy (Acc) [14], as given by Equation 7, 8, 9, 10, and 11.

$$P = \frac{TP}{FP + TP}. \tag{7}$$

$$R = \frac{TP}{FN + TP}. \tag{8}$$

$$F = \frac{2 \times R \times P}{R + P}. \tag{9}$$

$$AUC = \sum_{i=0}^{1} \frac{P(R_{i-1}) + P(R_i)}{2} \times \Delta R_i, \qquad \Delta R_i = R_i - R_{i-1}. \tag{10}$$

$$Acc = \frac{TP + TN}{FP + TP + FN + TN}. \tag{11}$$

Where, TP is the number of true positive predictions, FP is the number of false positives, FN is the number of false negatives, and i represents the segmentation threshold. As common in RVS literature, P, R, and F values are given based on the optimal segmentation threshold calculated to achieve the best F1 score. To calculate the AUC, we use the Scikit-learn python library [56].

P is the proportion of correctly segmented pixels. R measures the proportion of ground truth pixels correctly segmented. P and R are interdependent. This makes it challenging to improve one without affecting the other. Also, it is difficult to establish true segmentation labels. Therefore, R is typically used, not P. The accuracy metric Acc might not be reliable in cases where the data has a significant class imbalance. In the case of RVS data, there are far more pixel values that represent the background (true negative, TN) than pixel values that represent the blood vessels (true positive, TP), making it a classic example of class imbalance [6]. In fact, we can observe in our results (see, for example, Figure 4.5) that there is very little difference between the Acc values of different RVS models. For the reasons mentioned above, the F1 score and AUC are considered more reliable performance indicators as they combine both P and R in a more unified way.

4.2 In-dataset Analysis and Model Training

In this section, based on in-dataset analysis, we examine which are the best RVS models, check if they overfit due to small dataset sizes, and examine if our VIPs method helps reduce such overfitting. We then perform 10-fold cross-validation and combine all three datasets into one to examine how RVS models perform under a larger dataset (of images having different sizes, quality, and acquisition devices).

We performed in-dataset analysis for all five baseline models on all three datasets, i.e., CHASEDB [2], DRIVE [1], and HRF [3], as shown in Table 4.1. We notice from the average values of Table 4.1 that overall, the best baseline is LWNet [6] followed by SegRNet [7]. Additionally, based on Table 4.2, we infer that LWNet [6] is the most compact model architecture while SegRNet [7] has approximately ten times the number of trainable parameters in comparison to LWNet [6], while the most computationally complex model IterNet [8] is approximately 200 times more complex.

	CHASEDB [2]					DRIVE [1]					HRF [3]				
Model	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
IterNet [8]	79.51	82.01	80.10	96.21	63.2	81.00	81.21	81.08	96.52	61.35	79.28	78.58	78.75	94.01	61.02
WANet [36]	88.71	75.68	81.68	97.39	64.32	86.35	78.23	82.09	95.34	63.21	67.11	78.65	72.42	93.13	62.65
FRUNet [12]	76.63	86.82	81.35	97.05	64.21	85.39	80.63	82.72	97.07	64.31	79.11	83.47	81.23	97.11	62.55
SegRNet [7]	82.41	85.29	83.83	97.81	64.04	82.06	83.46	82.27	96.91	64.08	80.06	80.4	80.23	95.96	65.35
LWNet [6]	85.50	86.16	85.83	97.78	66.52	81.50	82.79	82.14	97.05	63.81	79.02	80.84	79.92	96.74	64.22

	Average				
Model	P	R	F	Acc	AUC
IterNet [8]	79.93	80.60	79.98	95.58	61.86
WANet [36]	80.72	77.52	78.73	95.29	63.39
FRUNet [12]	80.38	83.64	81.77	97.08	63.69
SegRNet [7]	81.51	83.05	82.11	96.89	64.49
LWNet [6]	82.01	83.26	82.63	97.19	64.85

Table 4.1: Results for in-dataset analysis of RVS baselines. Red indicates best, blue second best, and green third best result.

Model	number of trainable parameters
IterNet [8]	13,604,072
WANet [36]	3,919,682
FRUNet [12]	5,719,621
SegRNet [7]	641,601
LWNet [6]	68,482

Table 4.2: Complexity of RVS models in terms of number of trainable parameters.

We select LWNet [6], SegRNet [7], and IterNet [8] to analyze the performance of heavy and light weight models, and the generalization capability of our VIPs method when validated using models of different computational complexity. These RVS models have their own unique set of data augmentation operations and modes (i.e., online for LWNet [6], and IterNet [8] and offline for SegRNet [7]). Recall that LWNet [6] and IterNet [8] have no preprocessing step, but SegRNet [7] does (which uses CLAHE for contrast enhancement and gamma correction; we term it CLAHEGC).

Table 4.3 indicates that due to the use of offline augmentation in SegRNet [7] the total number of images used for a single training epoch is 12 times that of LWNet [6], additionally since our VIPs is an online augmentation method it has no effect on the number of images used in each training epoch.

	Number of training images in a single Epoch		
Model	CHASEDB [2] (20 original training samples)	DRIVE [1] (20 original training samples)	HRF [3] (15 original training samples)
SegRNet [7]	260	260	195
LWNet [6]	20	20	15
SegRNet [7] + VIPs	260	260	195
LWNet [6] + VIPs	20	20	15

Table 4.3: Total number of training images used by LWNet [6] and SegRNet [7] during a single training epoch after their augmentations both with and without our VIPs.

However, as shown by Table 4.4, the total number of unique samples used by LWNet [6] during the entire training process is much larger than SegRNet [7]. This is because LWNet [6] performs training in an online fashion. Additionally, by adding our online VIPs to SegRNet [7], it gets 50 times the number of unique samples during the course of training.

	Number of unique samples seen during training		
Model	CHASEDB [2]	DRIVE [1]	HRF [3]
SegRNet [7]	260	260	195
LWNet [6]	800	400	450
SegRNet [7] + VIPs	13000	13000	9750
LWNet [6] +VIPs	800	400	450

Table 4.4: Total number of unique image samples seen during the entire training process by LWNet [6] and SegRNet [7] after augmentations both with and without our VIPs.

LWNet [6], SegRNet [7], and IterNet [8] use Adam optimizer [57] to obtain the update weights as given by Equation 12.

$$\begin{aligned} w_{i+1} &= w_i - \frac{m_i}{1-\beta_2^i} \times (\frac{\alpha}{\sqrt{\frac{v_i}{1-\beta_2^i}}} + \epsilon), \\ m_i &= \beta_1 \times m_{i-1} + (1-\beta_1)(\frac{\delta\phi}{\delta w_i}), \\ v_i &= \beta_2 \times m_{i-1} + (1-\beta_2)(\frac{\delta\phi}{\delta w_i})^2. \end{aligned} \tag{12}$$

Where, w_i are the current weights at epoch i, w_{i+1} are the updated weights, α is the learning rate, β_1 and β_2 are the decay rate of the gradients moving average set at 0.9 and 0.99, m_i is the moving average of the gradients, v_i is the moving average of the squares of the gradients.

LWNet [6] uses two mini-U-nets cascaded in series, thus its overall loss function ϕ_t is a sum of the individual mini-U-net's binary cross entropy loss ϕ_1 and ϕ_2 as given in Equation 13.

$$\phi_t = \phi_1 + \phi_2. \tag{13}$$

Similarly, IterNet [8] cascades a U-net [4] into two mini U-nets connected in series and its overall loss function is also a sum of the individual U-net's [4] and mini U-nets binary cross entropy loss ϕ_1, ϕ_2, and ϕ_3 as shown in Equation 14.

$$\phi_t = \phi_1 + \phi_2 + \phi_3. \tag{14}$$

The individual binary cross entropy loss ϕ can be expressed using Equation 15.

$$\phi = \frac{1}{N} \times \sum_{i=1}^{N} (-(y_i \times \log(p_i)) + (1 - y_i) \times \log(1 - p_i)). \tag{15}$$

Where, N is the total number of predictions, y_i are the actual labels that are 0 or 1, and p_i are the predicted probabilities.

SegRNet [7] uses dice loss [58] (ϕ_d) as the loss function given in Equation 16.

$$\phi_d = 1 - \frac{\sum_{n=1}^{N} p_i \times y_i + \epsilon}{\sum_{n=1}^{N} p_i + y_i + \epsilon} - \frac{\sum_{n=1}^{N} (1 - p_i) \times (1 - y_i) + \epsilon}{\sum_{n=1}^{N} 2 - p_i - y_i + \epsilon}. \tag{16}$$

Where, N is the total number of predictions, y_i are the actual labels that are 0 or 1, p_i are the predicted probabilities, and ϵ is a very small value that prevents the denominator from ever becoming 0.

To examine the generalization capability of each baseline and how our VIPs method affects their generalization, we analyze the loss function plots at training and validation. Based on the training graphs of light-weight baseline LWNet [6] using the loss function in Equation 13, we observe in Figure 4.1 overfitting of baseline (black versus green curves) but our proposed VIPs (blue versus red) not only helps to reduce the training and validation loss but also reduces the gap between training and validation loss on all three datasets (DRIVE [1], CHASEDB [2], HRF [3]). This implies that our VIPs is useful in reducing over-fitting during the training process.

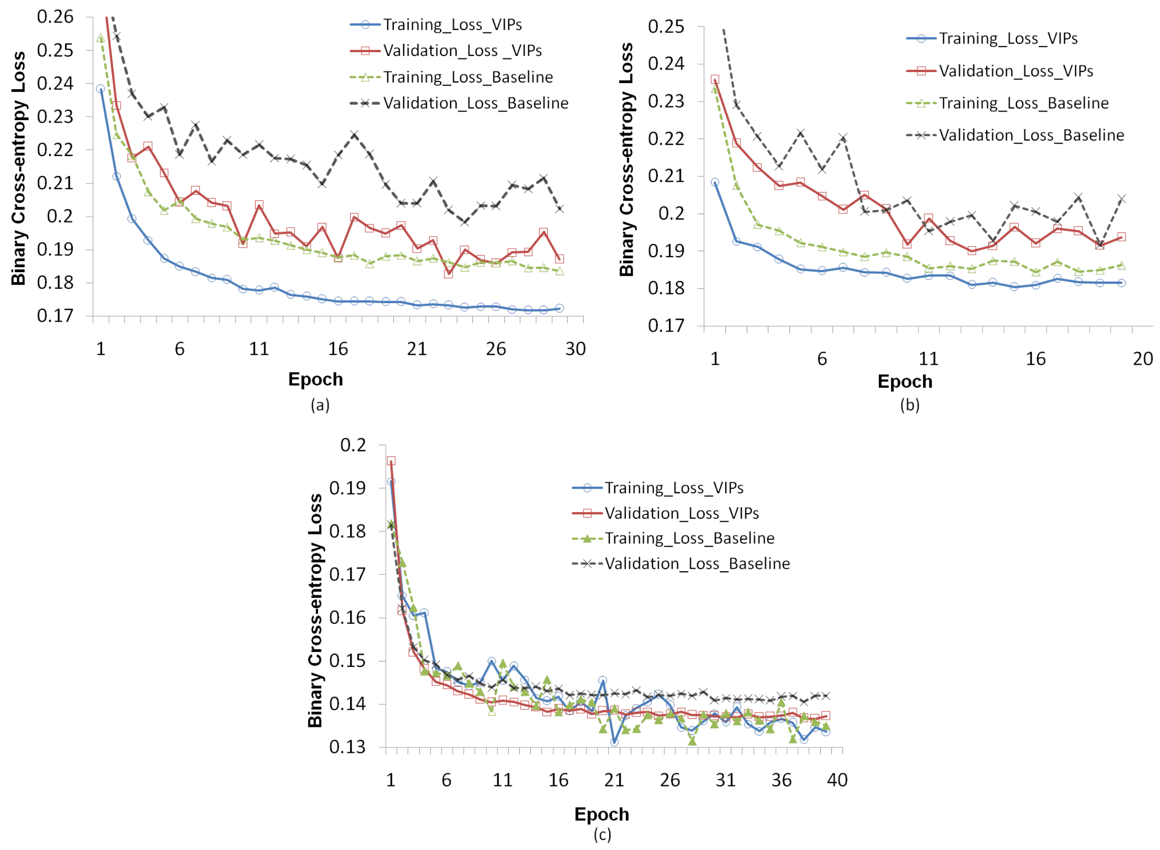

Figure 4.1: The training and validation loss as per Equation 13 versus the epoch for LWNet [6] trained and validated with and without VIPs (a) on HRF [3], (b) on DRIVE [1], and (c) on CHASEDB [2]. (CLAHEGC was used for preprocessing.) The number of epochs for each dataset is specified by the authors of the LWNet [6].

Based on the training plots of the more computationally complex SegRNet [7] as shown in Figure 4.2, we observe that the gaps between the training and validations loss are much larger than observed in LWNet [6] indicating that SegRNet [7] shows a higher tendency to overfitting on the training data. Figure 4.2 shows that using our VIPs reduces the gap between the training and validation losses across all three datasets (CHASEDB [2], DRIVE [1], and HRF [3]), indicating that using our VIPs helps to reduce overfitting.

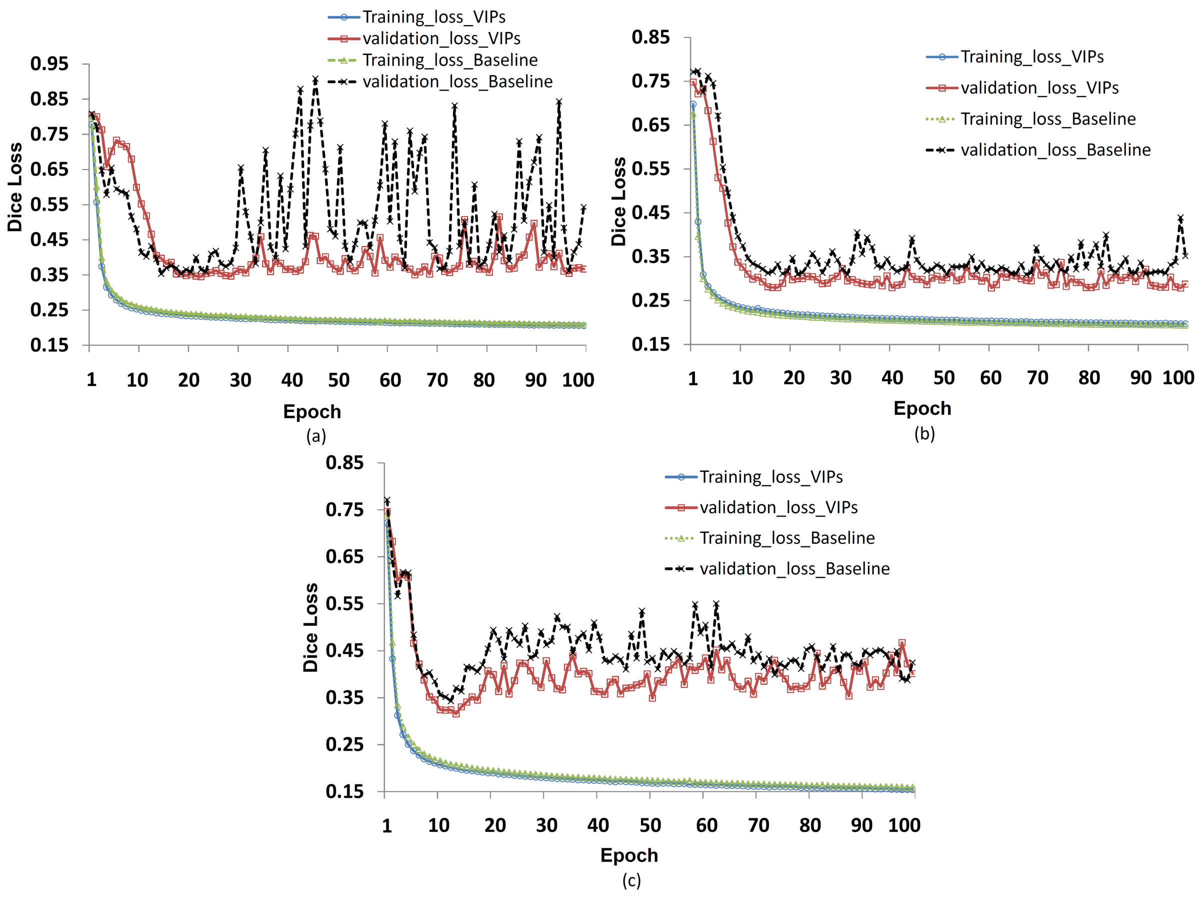

Figure 4.2: The training and validation loss as per Equation 16 versus the epoch for SegRNet [7] trained and validated with and without VIPs (a) on HRF [3], (b) on DRIVE [1], and (c) on CHASEDB [2]. (CLAHEGC was used for preprocessing.)

The training plots of IterNet [8] as shown in Figure 4.3 also have a larger gap between the training and validation losses than LWNet [6] and that they are significantly reduced across all three datasets when we use our VIPs. Based on our analysis of the training plots and the in-datasets analysis, we deduce 1) due to small dataset sizes, RVS models seem to suffer from the problem of over-fitting and using smaller, less complex models such as LWNet [6] seems much more beneficial than using more complex models with a larger number of trainable parameters, 2) despite the similar performance of SegRNet [7] and LWNet [6] in terms of F-score and AUC, in terms of model complexity and generalization LWNet [6] is a much better baseline model, and 3) using proposed VIPs reduces overfitting for all three models across all three datasets.

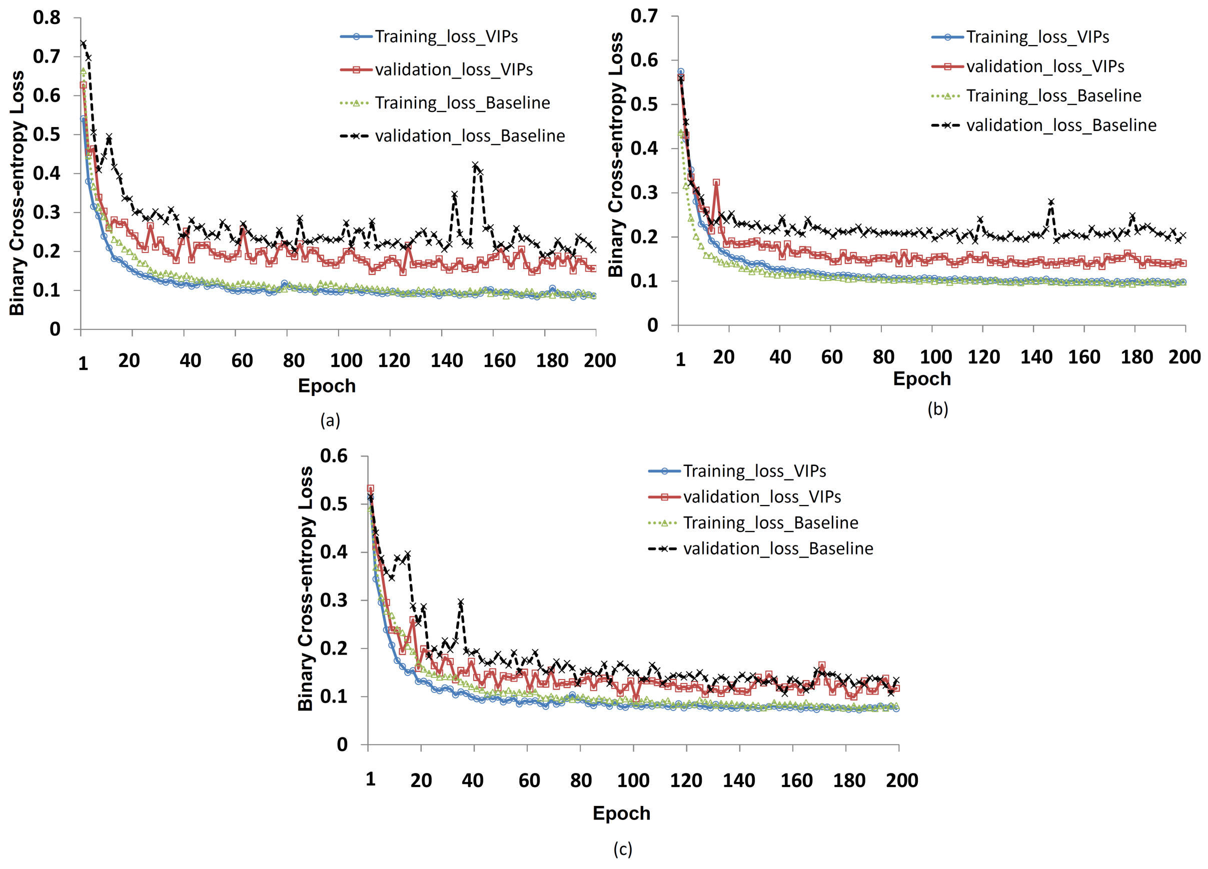

Figure 4.3: The training and validation loss as per Equation 14 versus the epoch for IterNet [8] trained and validated with and without VIPs (a) on HRF [3], (b) on DRIVE [1], and (c) on CHASEDB [2]. (CLAHEGC was used for preprocessing.)

To better analyze and validate the effects of our approach (VIPs+preprocessing) on RVS model generalization, we performed 10-fold cross-validation, both with and without our proposed VIPs on all three datasets (i.e., CHASEDB [2], DRIVE [1], and HRF [3]) using the best RVS model LWNet [6]. Based on the average values of Table 4.5, we observe that our VIPs method not only improves the performance of the baseline LWNet [6] but also reduces the standard deviation in all metrics.

	CHASEDB [2]					DRIVE [1]					HRF [3]				
Model	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
LWNet [6]	83.44	82.04	82.65	97.37	66.39	83.21	74.11	78.34	95.93	64.83	86.86	66.71	73.48	96.26	64.83
	(±3.18)	(±2.34)	(±0.94)	(±0.23)	(±0.45)	(±2.42)	(±2.69)	(±1.41)	(±0.23)	(±1.06)	(±4.37)	(±4.36)	(±4.93)	(±0.59)	(±3.23)
LWNet [6] + VIPs	84.73	81.23	82.90	97.44	66.17	84.05	74.12	78.71	96.02	65.183	85.08	69.39	76.29	96.27	65.83
	(±2.41)	(±1.77)	(±0.93)	(±0.23)	(±0.35)	(±2.43)	(±1.67)	(±0.88)	(±0.19)	(±0.43)	(±3.89)	(±4.22)	(±2.32)	(±0.61)	(±0.46)

	Average				
Model	P	R	F	Acc	AUC
LWNet [6]	84.50	74.29	78.16	96.52	65.35
	(±3.32)	(±3.13)	(±2.42)	(±0.35)	(±1.58)
LWNet [6] + VIPs	84.60	74.91	79.30	96.58	65.73
	(±2.91)	(±2.55)	(±1.37)	(±0.34)	(±0.41)

Table 4.5: Results for 10-fold cross validation of LWNet [6] both with and without VIPs + preprocessing. We give the standard deviations in ().

To study the effects of VIPs on a larger dataset with more diverse examples, we train and validate the best model, that is, LWNet [6] with and without VIPs on a combined dataset (we termed COMB) we created by combining the train and *test* splits of all three datasets (DRIVE [1], CHASEDB [2], and HRF [3]) for training and validation respectively. Recall these datasets, which have images of different aspect ratios captured using different imaging devices. Table 4.6 shows that our proposed VIPs improves LWNet [6] in all metrics and specifically by 1.61% in terms of F-score and 1.63% in terms of AUC.

	Combined Dataset				
Model	P	R	F	Acc	AUC
LWNet [6]	69.87	70.33	70.10	96.03	64.86
LWNet [6] +VIPs	70.10	73.40	71.71	96.84	66.49
	(0.23)	(3.06)	(1.61)	(0.80)	(1.63)

Table 4.6: Results for in-dataset analysis of LWNet [6] both with and without VIPs + preprocessing on combined dataset. We give the gains/losses over the baseline in ().

To examine how the model performs under different datasets, we compare the validation F-score and AUC values versus the size of the training sets (from small to larger) in Figure 4.4. First, we observe that using our proposed VIPs + preprocessing helps improve the validation F-score and AUC values for all training set sizes. Second, we see that combining the datasets into a large one "COMB" does not seem to be helpful. This can be due to the variable quality of the images in the COMB dataset. A word of caution is that these results are in-dataset. Cross-dataset comparison (as given in the next sections) is more reliable.

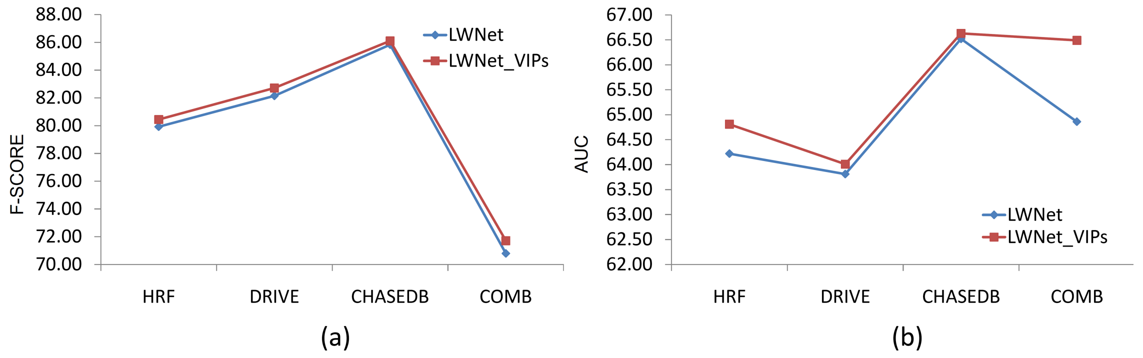

Figure 4.4: Validation scores versus the size of training sets both with and without VIPs. (a) F-score and (b) AUC. (CLAHEGC was used for preprocessing.)

4.3 Cross: Comparison of Baseline RVS Models

Due to the small size of the RVS datasets but also due lack of train, validate, an test splits, a cross-dataset analysis allows to evaluate model's performance with more confidence than in-dataset analysis. Models are evaluated with unseen data from a fully different dataset. We systematically considered all six possible cross-combinations of the three datasets, i.e., DRIVE [1], CHASEDB [2], and HRF [3], selected as shown in Table 4.7. We notice from the average values of Table 4.7 and Figure 4.7 that overall, the best baseline is LWNet [6] followed by SegRNet [7]. In the following analysis we thus, focus on studying these two RVS models.

	DRIVE [1] to CHASEDB [2]					DRIVE [1] to HRF [3]					CHASEDB [2] to DRIVE [1]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
WANet [36]	74.51	33.24	45.97	94.00	35.45	65.96	61.57	63.69	93.47	54.58	88.02	56.53	68.84	95.51	56.23
FRUNet [12]	68.08	28.72	40.21	94.91	36.49	4.08	55.03	7.57	47.91	15.66	89.16	54.93	67.98	94.32	49.55
IterNet [8]	52.24	51.68	51.96	93.90	36.42	54.56	60.22	57.25	93.42	32.97	63.87	63.61	63.74	92.15	35.77
SegRNet [7]	63.25	81.89	71.37	95.44	66.88	46.90	88.21	61.24	93.73	54.30	57.86	83.49	68.35	94.41	45.63
LWNet [6]	74.10	80.73	77.28	96.35	70.60	66.63	70.87	68.68	95.12	58.50	80.05	80.11	80.18	96.58	53.72

	CHASEDB [2] to HRF [3]					HRF [3] to DRIVE [1]					HRF [3] to CHASEDB [2]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
WANet [36]	77.51	77.93	77.72	95.55	63.54	43.32	69.48	53.37	89.35	36.65	55.98	70.18	62.28	94.13	54.58
FRUNet [12]	5.49	74.86	10.37	47.91	27.21	85.65	59.84	70.08	95.63	44.37	73.28	64.75	68.56	96.56	52.91
IterNet [8]	77.72	71.55	74.51	96.81	54.91	74.14	77.77	75.91	93.52	49.50	57.87	53.91	55.82	92.31	41.23
SegRNet [7]	53.62	83.04	65.16	94.63	58.32	64.56	77.10	70.24	94.55	50.23	76.48	47.86	58.88	94.99	48.30
LWNet [6]	67.36	74.78	70.88	95.74	60.29	71.95	74.77	73.33	95.07	53.19	58.54	57.02	57.77	91.32	43.05

	Average				
Method	P	R	F	Acc	AUC
WANet [36]	67.55	61.49	61.98	93.67	50.17
FRUNet [12]	54.29	56.36	44.13	79.54	37.70
IterNet [8]	63.40	63.12	63.20	93.69	41.80
SegRNet [7]	60.45	76.93	65.87	94.63	53.94
LWNet [6]	69.77	73.05	71.35	95.03	56.56

Table 4.7: Results for cross-dataset analysis of RVS baselines. Red indicates best, blue second best, and green third best result. Note the difference between best (LWNet [6]) and second (SegRNet [7]) best model in *AUC* is 2.62% and in F-score is 5.48%.

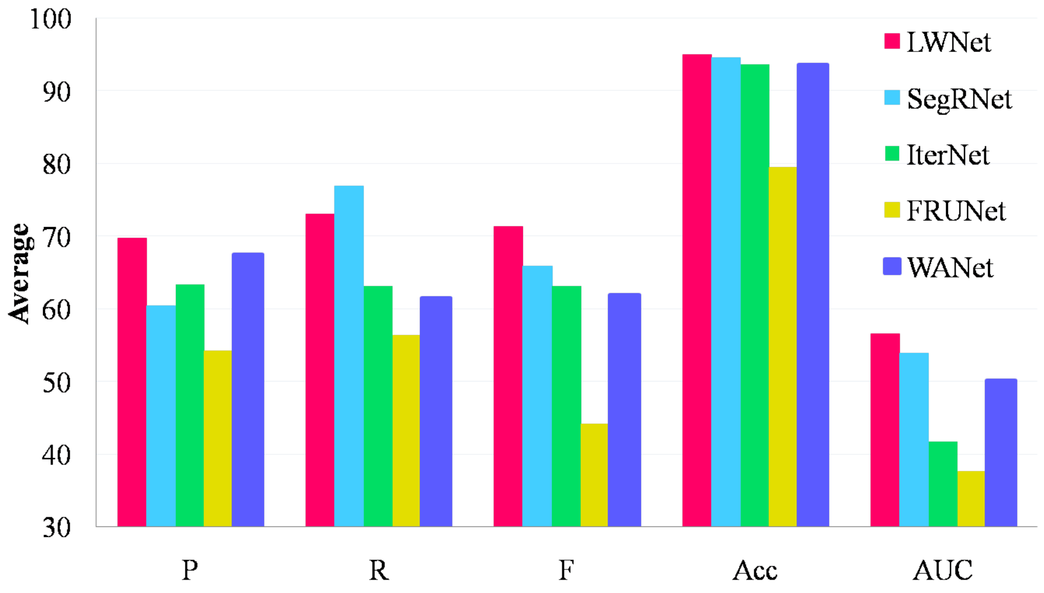

Figure 4.5: Graphical representation of the average P, R, F, Acc, and AUC values for the cross-dataset analysis of RVS baselines as shown in the last part of Table 4.7.

4.4 Cross-dataset: Impact of Components of RVS

Impact of preprocessing: There is little research on preprocessing methods for RVS. SegRNet [7] proposes a preprocessing method for RVS. The paper LIOT [22] proposes a preprocessing method for the segmentation of curvilinear objects (cracks

	DRIVE [1] to CHASEDB [2]					DRIVE [1] to HRF [3]					CHASEDB [2] to DRIVE [1]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
Preprocessing															
LWNet [6] with	56.13	71.79	63.01	94.89	55.53	68.13	70.32	69.21	95.08	63.99	70.74	75.95	73.25	95.14	54.76
LIOT [22]	(-17.97)	(-8.94)	(-14.27)	(-1.46)	(-15.07)	(1.50)	(-0.55)	(0.53)	(-0.04)	(5.49)	(-9.31)	(-4.16)	(-6.93)	(-1.44)	(1.04)
LWNet [6] with	76.55	82.06	79.21	96.52	70.61	70.55	68.39	69.45	96.06	56.69	80.69	79.05	79.86	96.12	52.26
CLHAEGC	(2.45)	(1.33)	(1.93)	(0.17)	(0.01)	(3.92)	(-2.48)	(0.77)	(0.94)	(-1.81)	(0.64)	(-1.06)	(-0.32)	(-0.46)	(-1.46)
SegRNet [7] without	70.26	58.89	64.07	95.41	57.02	58.28	77.47	66.52	96.24	49.26	82.43	45.08	58.28	96.44	45.87
CLHAEGC	(7.01)	(-23.00)	(-7.30)	(-0.03)	(-9.86)	(11.38)	(-10.74)	(5.28)	(2.51)	(-5.04)	(24.57)	(-38.41)	(-10.07)	(2.03)	(0.24)
No data augmentation (DA)															
LWNet [6] no DA	51.16	53.26	52.19	92.50	35.51	71.13	69.43	70.27	96.28	59.83	78.66	75.15	76.87	96.01	52.97
	(-22.94)	(-27.47)	(-25.09)	(-3.85)	(-35.09)	(4.50)	(-1.44)	(1.59)	(1.16)	(1.33)	(-1.39)	(-4.96)	(-3.31)	(-0.57)	(-0.75)
SegRNet [7] no DA	97.75	0.31	0.61	93.08	5.43	93.05	3.66	7.05	94.58	6.27	93.55	11.00	19.69	94.71	12.40
	(34.50)	(-81.58)	(-70.76)	(-2.36)	(-61.45)	(46.15)	(-84.55)	(-54.19)	(0.85)	(-48.03)	(35.69)	(-72.49)	(-48.66)	(0.30)	(-33.23)
Augmentation swap															
LWNet [6] with	32.80	67.50	44.15	89.85	33.21	70.07	64.05	66.92	96.08	62.59	77.40	76.94	77.17	96.01	57.13
SegRNet's [7] DA	(-41.30)	(-13.23)	(-33.13)	(-6.50)	(-37.39)	(3.44)	(-6.82)	(-1.76)	(0.96)	(4.09)	(-2.65)	(-3.17)	(-3.01)	(-0.57)	(3.41)
SegRNet [7] with	30.13	26.85	28.40	92.18	23.48	47.17	36.08	40.89	95.45	50.02	72.55	71.45	71.99	95.79	50.56
LWNet's [6] DA	(-33.12)	(-55.04)	(-42.97)	(-3.26)	(-43.40)	(0.27)	(-52.13)	(-20.35)	(1.72)	(-4.28)	(14.69)	(-12.04)	(3.64)	(1.38)	(4.93)
Online versus offline															
LWNet [6] offline	64.94	78.63	71.13	91.80	38.32	74.37	80.92	77.51	96.20	59.71	77.22	78.78	77.99	96.17	59.71
with its DA	(-9.16)	(-2.10)	(-6.15)	(-4.55)	(-32.28)	(7.74)	(10.05)	(8.83)	(1.08)	(1.21)	(-2.83)	(-1.33)	(-2.19)	(-0.41)	(5.99)
SegRNet [7] online	64.79	27.45	38.57	93.93	34.32	57.70	60.71	59.17	95.29	52.65	66.96	66.52	66.74	93.34	49.39
with its DA	(1.54)	(-54.44)	(-32.80)	(-1.51)	(-32.56)	(10.80)	(-27.50)	(-2.07)	(1.56)	(-1.65)	(9.10)	(-16.97)	(-1.61)	(-1.07)	(3.76)

	CHASEDB [2] to HRF [3]					HRF [3] to DRIVE [1]					HRF [3] to CHASEDB [2]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
Preprocessing															
LWNet [6] with	70.97	71.89	71.43	95.48	58.46	66.64	73.32	69.82	94.44	56.31	59.54	65.78	62.51	95.22	55.03
LIOT [22]	(3.61)	(-2.89)	(0.55)	(-0.26)	(-1.83)	(-5.31)	(-1.45)	(-3.51)	(-0.630)	(3.12)	(1.00)	(8.76)	(4.74)	(3.90)	(11.98)
LWNet [6] with	76.12	69.61	72.72	96.65	56.79	77.55	77.84	77.70	95.36	60.15	70.90	69.72	70.31	94.22	49.03
CLAHEGC	(8.76)	(-5.17)	(1.84)	(0.91)	(-3.50)	(5.60)	(3.07)	(4.37)	(0.29)	(6.96)	(12.36)	(12.70)	(12.54)	(2.90)	(5.98)
SegRNet [7] without	81.97	48.06	60.59	96.44	51.87	80.12	42.28	55.35	96.24	49.26	89.13	0.05	10.60	93.40	9.43
CLAHEGC	(28.35)	(-34.98)	(-4.57)	(1.81)	(-6.45)	(15.56)	(-34.82)	(-14.89)	(1.69)	(-0.97)	(12.65)	(-47.80)	(-48.28)	(-1.59)	(-38.87)
No data augmentation (DA)															
LWNet [6] no DA	74.76	70.55	72.59	96.53	59.27	71.26	73.33	72.28	95.05	55.63	43.74	45.43	44.57	88.73	23.00
	(7.40)	(-4.23)	(1.71)	(0.79)	(-1.02)	(-0.69)	(-1.44)	(-1.05)	(-0.02)	(2.44)	(-14.8)	(-11.59)	(-13.20)	(-2.59)	(-20.05)
SegRNet [7] no DA	68.00	7.76	13.93	94.96	17.52	9.33	50.69	15.76	70.12	14.02	13.64	34.65	19.57	80.23	17.42
	(14.38)	(-75.28)	(-51.23)	(0.33)	(-40.80)	(-55.23)	(-26.41)	(-54.48)	(-24.43)	(-36.21)	(-62.84)	(-13.21)	(-39.31)	(-14.76)	(-30.88)
Augmentation swap															
LWNet with SegRNet DA	71.78	72.85	72.31	96.54	59.88	71.77	72.83	72.30	95.11	63.47	31.37	22.50	26.20	91.36	11.32
SegRNet's [7] DA	(4.42)	(-1.93)	(1.43)	(0.80)	(-0.41)	(-0.18)	(-1.94)	(-1.03)	(0.04)	(10.28)	(-27.17)	(-34.52)	(-31.57)	(0.04)	(-31.73)
SegRNet with	59.57	66.68	62.93	94.67	55.29	64.80	59.85	62.23	95.50	46.87	43.43	40.17	41.74	93.76	40.03
LWNet's [6] DA	(5.95)	(-16.36)	(-2.23)	(0.04)	(-3.03)	(0.24)	(-17.25)	(-8.01)	(0.95)	(-3.36)	(-33.05)	(-7.69)	(-17.14)	(-1.23)	(-8.27)
Online versus offline															
LWNet [6] offline	62.76	65.01	63.86	96.51	58.75	55.59	58.03	56.78	92.42	24.05	71.25	69.90	70.57	95.21	58.61
with its DA	(-4.60)	(-9.77)	(-7.02)	(0.77)	(-1.54)	(-16.36)	(-16.74)	(-16.55)	(-2.65)	(-29.14)	(12.71)	(12.88)	(12.80)	(3.89)	(15.56)
SegRNet [6]online	62.68	56.00	59.15	95.66	52.64	75.77	30.47	43.47	94.22	48.70	65.53	66.39	65.96	94.49	38.68
with its DA	(9.06)	(-27.04)	(-6.01)	(1.03)	(-5.68)	(11.21)	(-46.63)	(-26.77)	(-0.33)	(-1.53)	(-10.95)	(18.53)	(7.08)	(-0.50)	(-9.62)

	Average				
Method	P	R	F	Acc	AUC
Preprocessing					
LWNet [6] with	65.36	71.51	68.21	95.04	57.35
LIOT [22]	(-4.41)	(-1.54)	(-3.15)	(0.01)	(0.79)
LWNet [6] with	75.39	74.45	74.88	95.82	57.59
CLAHEGC	(5.62)	(1.40)	(3.52)	(0.79)	(1.03)
SegRNet [7] without	77.03	45.31	52.57	95.70	43.79
CLAHEGC	(16.59)	(-31.63)	(-13.31)	(1.07)	(-10.16)
No data augmentation (DA)					
LWNet [6] no DA	65.12	64.53	64.80	94.18	47.70
	(-4.65)	(-8.52)	(-6.56)	(-0.85)	(-8.86)
SegRNet [6] no DA	62.55	18.01	12.77	87.95	12.18
	(2.11)	(-58.92)	(-53.11)	(-6.68)	(-41.77)
Augmentation swap					
LWNet [6] with	59.20	62.78	59.84	94.16	47.93
SegRNet's [7] DA	(-10.57)	(-10.27)	(-11.51)	(-0.87)	(-8.63)
SegRNet [7] with	52.94	50.18	51.36	51.36	51.36
LWNet's [6] DA	(-7.50)	(-26.75)	(-14.51)	(-0.07)	(-9.57)
Online versus offline					
LWNet [6] offline	67.69	71.88	69.64	94.72	49.86
with its DA	(-2.08)	(-1.17)	(-1.71)	(-0.31)	(-6.70)
SegRNet [7] online	65.57	51.26	55.51	94.49	46.06
with its DA	(5.13)	(-25.68)	(-10.36)	(-0.14)	(-7.88)

Table 4.8: Impact of augmentation and preprocessing modalities on LWNet [6] and SegRNet [7]. We give the gains/loss over baseline in ().

and retinal vessels). We compared the preprocessing of SegRNet [7] (we term it CLAHEGC) and LIOT [22] for RVS. As seen in the preprocessing part of Table 4.8, CLAHGC is by far superior to LIOT [22]. Comparing with and without preprocessing, Table 4.8 shows a significant performance drop of -4.57 % in the F-scores for SegRNet [7] when we train it without its CLAHEGC preprocessing. While adding CLAHEGC to LWNet [6] gives average gains of 3.52 % in F-score. These results indicate that adequate preprocessing of the data before training benefits RVS models.

Impact of not using data augmentation: From the third part of Table 4.8, it is evident that RVS heavily depends on data augmentation techniques; we observe average losses of -6.55 % and -53.10 % in terms of F-score for LWNet [6] and SegRNet [59] respectively when no data augmentation is used.

Impact of unique data augmentation: Through this experiment, we aim to understand the dependency of an RVS model on its own data augmentation. From the fourth part of Table 4.8, we conclude that the specific operations for augmentation proposed by the authors are an integral part of the RVS models in question: swapping the set of augmentation between LWNet [6] and SegRNet [7] leads to average losses of -14.51 % and -11.51 % in terms of F-score for LWNet [6] and SegRNet [7] respectively.

Impact of online versus offline augmentation: We change the mode of data augmentation for LWNet [6] from online to offline and vice-versa for SegRNet [7] while keeping their data augmentation as mentioned by their authors; this is done to understand the impact that augmentation modes have on model performance. In the fifth part of Table 4.8, we observe on average losses of -1.71 % and -10.36 % for LWNet [6] and SegRNet [7] indicating that the mode of augmentation is important and model specific.

Conclusion: based on the above studies, a given RVS model seems to operate optimally with its own set of operations for augmentation and its mode (online or offline). However, the set of preprocessing used for SegRNet [7] has shown significant gains for LWNet [6], while LIOT [22] is not useful for LWNet [36]. The losses caused due to the use of offline augmentation mode (that is, increasing the number of training samples before training) for LWNet [6] indicates that simply increasing the number of images blindly does not lead to improved model generalization. Adequate preprocessing (CLAHEGC) appears to be a crucial step that improves model generalization; one reason could be that preprocessing increases the contrast between the retinal vessels and background, thus reducing the chances of pixel miss-classification.

4.5 Proposed Versus Related Augmentation Works

Using cross-datasets, we study the performance of our VIPs augmentation method (with parameters p=0.5, α=0.4, β=0.5, and γ=10), the RVS-specific augmentation method [9] (we term as "Robust"), and the general-purpose augmentation methods Mosaic [23], CutMix [10], and Erase [17]. For this, based on our study in Section 4.4, we add them to the baselines for LWNet [6] as-is but equipped with CLAHEGC preprocessing and to SegRNet [7] as-is (with its CLAHEGC). The results are in Tables 4.9 and 4.10. Comparing the two tables, we conclude that our method is stable across RVS models and datasets and by far outperforms all related works.

	DRIVE [1] to CHASEDB [2]					DRIVE [1] to HRF [3]					CHASEDB [2] to DRIVE [1]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
LWNet [6] + VIPs	76.03	82.50	79.13	96.78	70.71	74.05	75.87	74.95	95.94	68.23	81.14	81.69	81.41	96.63	54.84
	(1.93)	(1.77)	(1.85)	(0.43)	(0.11)	(7.42)	(5.00)	(6.27)	(0.82)	(9.73)	(1.09)	(1.58)	(1.23)	(0.05)	(1.12)
LWNet [6] + Robust [9]	76.53	83.16	79.71	96.74	70.28	76.97	70.39	73.53	95.79	65.82	81.4	79.65	80.51	96.60	49.94
	(2.43)	(2.43)	(2.43)	(0.39)	(-0.32)	(10.34)	(-0.48)	(4.85)	(0.67)	(7.32)	(1.35)	(-0.46)	(0.33)	(0.02)	(-3.78)
LWNet [6] + Mosaic [23]	75.36	81.59	78.35	96.53	70.67	71.20	67.64	69.37	95.3	58.05	80.18	76.65	78.38	96.31	40.9
	(1.26)	(0.86)	(1.07)	(0.18)	(0.07)	(4.57)	(-3.23)	(0.69)	(0.18)	(-0.45)	(0.13)	(-3.46)	(-1.80)	(-0.27)	(-12.82)
LWNet [6] + cutmix [10]	75.52	84.11	79.59	96.76	71.23	72.29	73.06	72.67	95.62	63.95	80.52	78.78	79.64	96.49	44.62
	(1.42)	(3.38)	(2.31)	(0.41)	(0.63)	(5.66)	(2.19)	(3.99)	(0.5)	(5.45)	(0.47)	(-1.33)	(-0.54)	(-0.09)	(-9.10)
LWNet [6] + Erase [17]	75.86	81.32	78.50	96.57	72.61	71.27	65.90	68.48	95.16	58.19	80.94	77.14	78.99	96.47	48.27
	(1.76)	(0.59)	(1.22)	(0.22)	(2.01)	(4.64)	(-4.97)	(-0.20)	(0.04)	(-0.31)	(0.89)	(-2.97)	(-1.19)	(-0.11)	(-5.45)

	CHASEDB [2] to HRF [3]					HRF [3] to DRIVE [1]					HRF [3] to CHASEDB [2]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
LWNet [6]+ VIPs	76.66	76.56	76.61	96.25	60.61	78.21	77.46	77.83	96.06	63.26	71.63	67.71	69.61	95.46	61.78
	(9.30)	(1.78)	(5.73)	(0.51)	(0.32)	(6.26)	(2.69)	(4.50)	(0.99)	(10.07)	(13.09)	(10.69)	(11.84)	(4.14)	(18.73)
LWNet [6] + Robust [9]	76.97	70.39	73.53	95.95	57.70	75.89	77.98	76.92	95.80	64.98	69.58	66.78	68.15	95.31	60.76
	(9.61)	(-4.39)	(2.65)	(0.21)	(-2.59)	(3.94)	(3.21)	(3.59)	(0.73)	(11.79)	(11.04)	(9.76)	(10.38)	(3.99)	(17.71)
LWNet [6] + Mosaic [23]	76.42	66.33	71.02	95.66	48.37	76.49	79.10	77.78	96.04	47.56	64.73	71.29	67.85	94.81	45.89
	(9.06)	-(8.45)	(0.14)	(-0.07)	(-11.92)	(4.54)	(4.33)	(4.45)	(0.97)	(-5.63)	(6.19)	(14.27)	(10.08)	(3.49)	(2.84)
LWNet [6] + Cutmix [10]	76.77	71.08	73.82	95.97	55.07	76.35	78.16	77.24	95.96	50.32	69.43	71.39	70.39	95.38	55.14
	(9.41)	(-3.7)	(2.94)	(0.23)	(-5.22)	(4.40)	(3.39)	(3.91)	(0.89)	(-2.87)	(10.89)	(14.37)	(12.62)	(4.06)	(12.09)
LWNet [6] + Erase [17]	75.83	65.96	70.55	95.62	48.65	73.86	77.36	75.57	96.07	50.88	66.15	68.42	67.27	94.88	54.68
	(8.47)	(-8.82)	(-0.33)	(-0.11)	(-11.64)	(1.91)	(2.59)	(2.24)	(1.00)	(-2.31)	(7.61)	(11.40)	(9.50)	(3.56)	(11.63)

	Average				
Method	P	R	F	Acc	AUC
LWNet [6] + VIPs	76.29	76.97	76.59	96.19	63.24
	(6.51)	(3.91)	(5.23)	(1.15)	(6.68)
LWNet [6] + Robust [9]	76.22	74.73	75.39	96.03	61.58
	(6.45)	(1.67)	(4.03)	(1.00)	(5.02)
LWNet [6] + Mosaic [23]	74.06	73.77	73.79	95.78	51.91
	(4.29)	(0.72)	(2.43)	(0.74)	(-4.65)
LWNet [6] + Cutmix [10]	75.15	76.10	75.56	96.03	56.72
	(5.37)	(3.05)	(4.20)	(1.00)	(0.16)
LWNet [6] + Erase [17]	73.99	72.68	73.23	95.80	55.55
	(4.21)	(-0.36)	(1.87)	(0.76)	(-1.01)

Table 4.9: Comparing LWNet [6] with proposed VIPs and related augmentation techniques. CLAHEGC was used for preprocessing. We give the gains/loss over baseline in (). Red indicates best, blue second best, and green third best result respectively.

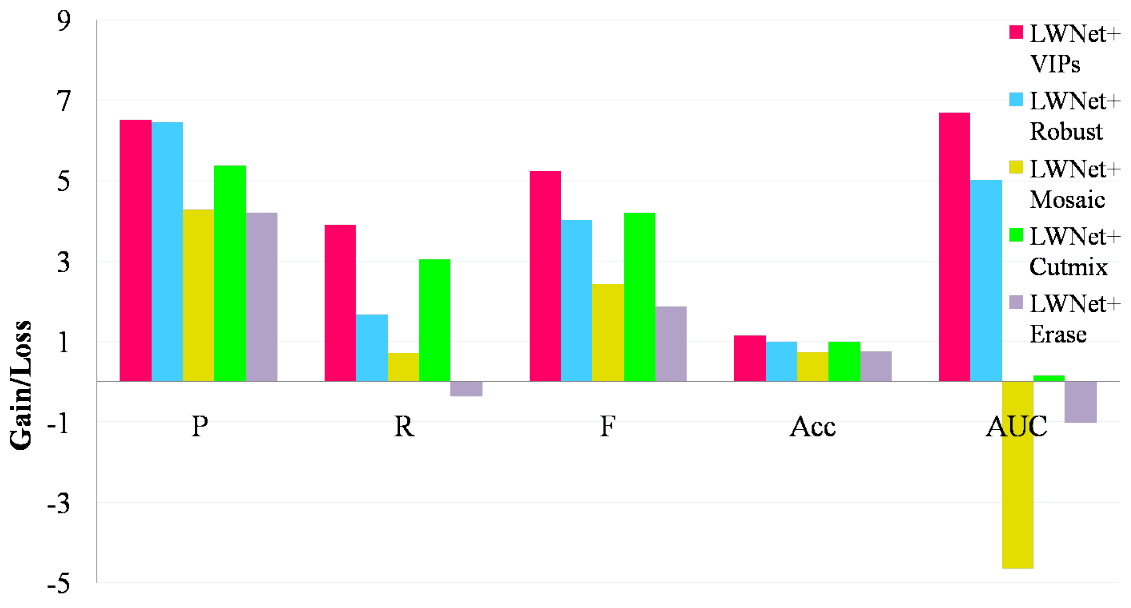

Figure 4.6: LWNet [6] (with CLAHEGC used for preprocessing): Graphical representation of the average gains/loss in Table 4.9, in P, R, F, Acc, and AUC comparing our proposed VIPs and related augmentation techniques.

For LWNet baseline [6], Table 4.9 and Figure 4.6 shows that our VIPs significantly improves the baseline on average by 6.68% in AUC and by 5.23% in F-score. Our method outperforms all related augmentation methods, e.g., it gives an average gain of 5.23 % in F-score versus 4.20% of the second best related work CutMix [10]. It is Interesting to note that our method introduces no loss in any of the six individual cross-dataset cases, whereas the second and third best methods, “CutMix” and “Robust” introduce major losses (e.g., -4.39% in R under CHASEDB [2] to HRF [3]).

For the baseline SegRNet [7], Table 4.10 and Figure 4.7 shows that our VIPs remains the best augmentation method, as it is for LWNet [6]. For instance, it gives best average gains of 5.29% in AUC and of 1.03% in F-score over baseline while all related methods cause major losses in F-score and the second best (Cutmix [10]) method is inferior to our method by AUC is 1.49%.

	DRIVE [1]to CHASEDB [2]					DRIVE [1] to HRF [3]					CHASEDB [2] to DRIVE [1]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
SegRNet [7] + VIPs	64.72 (1.47)	81.81 (-0.08)	72.27 (0.90)	95.66 (0.22)	68.69 (1.81)	48.53 (1.63)	87.53 (-0.68)	62.44 (1.20)	93.79 (0.06)	60.39 (6.09)	58.77 (0.91)	83.94 (0.45)	69.14 (0.79)	96.06 (1.65)	55.32 (9.69)
SegRNet [7] + Robust [23]	73.57 (10.32)	67.76 (-14.13)	70.55 (-0.82)	96.07 (0.63)	66.33 (-0.55)	57.56 (10.66)	80.99 (-7.22)	67.29 (6.05)	95.58 (1.85)	59.89 (5.59)	59.74 (1.88)	79.97 (-3.52)	68.39 (0.04)	95.92 (1.51)	57.46 (11.83)
SegRNet [7] + Mosaic [23]	72.36 (9.11)	68.82 (-13.07)	70.55 (-0.82)	96.01 (0.57)	66.31 (-0.57)	53.02 (6.12)	84.09 (-4.12)	65.03 (3.79)	94.92 (1.19)	57.87 (3.57)	75.53 (17.67)	56.32 (-27.17)	64.52 (-3.83)	96.58 (2.17)	54.19 (8.56)
SegRNet [7]+ Cutmix [10]	80.79 (17.54)	47.73 (-34.16)	60.01 (-11.36)	95.58 (0.14)	56.40 (-10.48)	62.80 (15.90)	72.16 (-16.05)	67.15 (5.91)	96.04 (2.31)	59.76 (5.46)	67.73 (9.87)	72.38 (-11.11)	69.98 (1.63)	96.57 (2.16)	58.78 (13.15)
SegRNet [7]+ Erase [17]	72.10 (8.85)	51.22 (-30.67)	59.89 (-11.48)	95.29 (-0.15)	56.29 (-10.59)	56.24 (9.34)	63.97 (-24.24)	59.85 (-1.39)	95.18 (1.45)	53.27 (-1.03)	72.96 (15.10)	62.66 (-20.83)	67.42 (-0.93)	96.66 (2.25)	56.62 (10.99)
	CHASEDB [2] to HRF [3]					HRF [3] to DRIVE [1]					HRF [3] to CHASEDB [2]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
SegRNet [7] + VIPs	54.58 (0.96)	82.99 (-0.05)	65.85 (0.69)	95.78 (1.15)	58.36 (0.04)	64.64 (0.08)	76.44 (-0.66)	70.05 (-0.19)	96.26 (1.71)	62.33 (12.10)	77.67 (1.19)	51.20 (3.34)	61.72 (2.84)	95.11 (0.12)	50.32 (2.02)
SegRNet [7] + Robust [9]	51.69 (-1.93)	77.84 (-5.20)	62.12 (-3.04)	94.67 (0.04)	55.29 (-3.03)	74.32 (9.76)	59.72 (-17.38)	66.23 (-4.01)	96.64 (2.09)	58.94 (8.71)	84.05 (7.57)	29.07 (-18.79)	43.20 (-15.68)	94.69 (-0.30)	40.60 (-7.70)
SegRNet [7] + Mosaic [23]	75.25 (21.63)	58.77 (-24.27)	66.00 (0.84)	96.60 (1.97)	58.73 (0.41)	53.85 (-10.71)	65.38 (-11.72)	59.06 (-11.18)	95.00 (0.45)	52.56 (2.33)	84.10 (7.62)	32.29 (-15.57)	46.67 (-12.21)	94.87 (-0.12)	43.86 (-4.44)
SegRNet [7]+ Cutmix [10]	64.09 (10.47)	72.78 (-10.26)	68.16 (3.00)	96.18 (1.55)	60.63 (2.31)	70.02 (5.46)	65.57 (-11.53)	67.72 (-2.52)	96.55 (2.00)	60.27 (10.04)	79.61 (3.13)	40.71 (-7.15)	53.87 (-5.01)	95.16 (0.17)	50.63 (2.33)
SegRNet [7] + Erase [17]	65.72 (12.10)	63.51 (-19.53)	64.60 (-0.56)	96.09 (1.46)	57.49 (-0.83)	66.43 (1.87)	59.14 (-17.96)	62.57 (-7.67)	96.10 (1.55)	55.69 (5.46)	71.75 (-4.73)	40.75 (-7.11)	51.98 (-6.90)	94.77 (-0.22)	48.85 (0.55)

	Average				
Method	P	R	F	Acc	AUC
SegRNet [7] + VIPs	61.49 (1.04)	77.32 (0.38)	66.91 (1.03)	95.44 (0.81)	59.24 (5.29)
SegRNet [7] + Robust [9]	66.82 (6.37)	65.89 (-11.04)	62.96 (-2.91)	95.60 (0.97)	56.42 (2.47)
SegRNet [7] + Mosaic [23]	69.02 (8.57)	60.95 (-15.98)	61.97 (-3.90)	95.66 (1.03)	55.59 (1.64)
SegRNet [7]+ Cutmix [10]	70.84 (10.39)	61.89 (-15.04)	64.48 (-1.39)	96.01 (1.38)	57.75 (3.80)
SegRNet [7] + Erase [17]	67.53 (7.08)	56.88 (-20.05)	61.05 (-4.82)	95.68 (1.05)	54.30 (0.75)

Table 4.10: Comparing SegRNet [7] with proposed VIPs and related augmentation techniques. CLAHEGC was used for preprocessing. We give the gains/loss over baseline in (). Red indicates best, blue second best, and green third best result respectively.

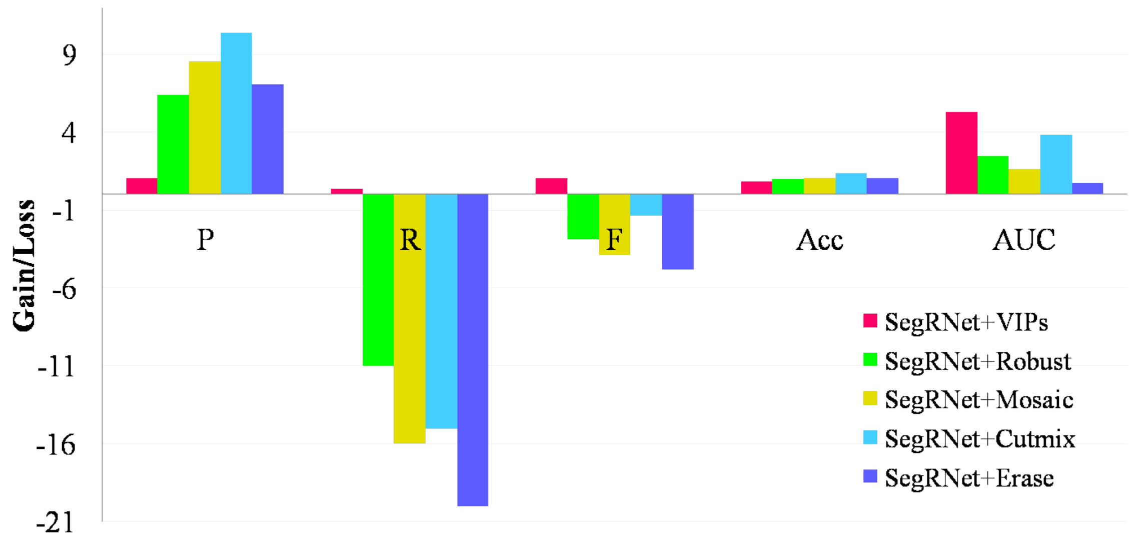

Figure 4.7: SegRNet [7]: Graphical representation of the average gains/loss in Table 4.10 in P, R, F, Acc, and AUC comparing proposed VIPs and related augmentation techniques.

4.6 Ablation study

In the first cross-dataset ablation study, we examine the effects of using preprocessing (CLAHEGC) only, data augmentation VIPs only, and data augmentation VIPs + preprocessing on the LWNet [6] baseline. For VIPs, we used the proposed set of hyper-parameters p=0.5, α=0.4, β=0.5, and γ=10. From Table 4.11 and Figure 4.8, we observe that adding "VIPs with preprocessing" to the LWNet [6] baseline gives the best results compared to "VIPs without preprocessing" and "preprocessing without VIPs". Not only is the average gain the highest among the three modes but "VIPs with preprocessing" is more stable, that is, it introduces no losses in any of the individual cross-dataset cases. The mode "preprocessing without VIPS" can introduce major losses (e.g., -5.17% in R under CHASEDB to HRF).

	DRIVE [1] to CHASEDB [2]					DRIVE [1] to HRF [3]					CHASEDB [2] to DRIVE [1]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
LWNet [6] + VIPs with preprocessing	76.03 (1.93)	82.50 (1.77)	79.13 (1.85)	96.78 (0.43)	70.71 (0.11)	74.05 (7.42)	75.87 (5.00)	74.95 (6.27)	95.95 (0.83)	68.23 (9.73)	81.14 (1.09)	81.69 (1.58)	81.41 (1.23)	96.63 (0.05)	54.84 (1.12)
LWNet [6] + VIPs without preprocessing	75.02 (0.92)	81.31 (0.58)	78.04 (0.76)	96.41 (0.06)	70.81 (0.21)	73.93 (7.30)	73.34 (2.47)	73.64 (4.96)	95.75 (0.63)	60.14 (1.64)	80.30 (0.25)	80.53 (0.42)	80.41 (0.23)	96.59 (0.01)	54.57 (0.85)
LWNet [6] with preprocessing (no VIPs)	76.55 (2.45)	82.06 (1.33)	79.21 (1.93)	96.52 (0.17)	70.61 (0.01)	70.55 (3.92)	68.39 (-2.48)	69.45 (0.77)	96.06 (0.94)	56.69 (-1.81)	80.69 (0.64)	79.05 (-1.06)	79.86 (-0.32)	96.12 (-0.46)	52.26 (-1.46)
	CHASEDB [2] to HRF [3]					HRF [3] to DRIVE [1]					HRF [3] to CHASEDB [2]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
LWNet [6] + VIPs with preprocessing	76.66 (9.30)	76.56 (1.78)	76.61 (5.73)	96.25 (0.51)	60.61 (0.32)	78.21 (6.26)	77.46 (2.69)	77.83 (4.50)	96.06 (0.99)	63.26 (10.07)	71.63 (13.09)	67.71 (10.69)	69.61 (11.84)	95.46 (4.14)	61.78 (18.73)
LWNet [6] + VIPs without preprocessing	76.79 (9.43)	75.13 (0.35)	75.95 (5.07)	96.17 (0.43)	61.03 (0.74)	74.33 (2.38)	75.78 (1.01)	75.05 (1.72)	95.58 (0.51)	63.27 (10.08)	61.24 (2.70)	58.76 (1.74)	59.97 (2.20)	93.97 (2.65)	47.38 (4.33)
LWNet [6] with preprocessing (no VIPs)	76.12 (8.76)	69.61 (-5.17)	72.72 (1.84)	96.65 (0.91)	56.79 (-3.50)	77.55 (5.60)	77.84 (3.07)	77.70 (4.37)	95.36 (0.29)	60.15 (6.96)	70.90 (12.36)	69.72 (12.70)	70.31 (12.54)	94.22 (2.90)	49.03 (5.98)

	Average				
Method	P	R	F	Acc	AUC
LWNet [6] + VIPs with preprocessing	76.29 (6.52)	76.97 (3.92)	76.59 (5.24)	96.19 (1.16)	63.24 (6.68)
LWNet [6] + VIPs without preprocessing	73.60 (3.83)	74.14 (1.10)	73.84 (2.49)	95.72 (0.72)	59.45 (2.98)
LWNet [6] with preprocessing (no VIPs)	75.39 (5.62)	74.45 (1.40)	74.88 (3.52)	95.82 (0.79)	57.59 (1.03)

Table 4.11: Ablation study: LWNet [6] with/without VIPs and with/without preprocessing (CLAHEGC). We give the gains/loss over baseline in (). Red indicates best, blue second best, and green third best result respectively.

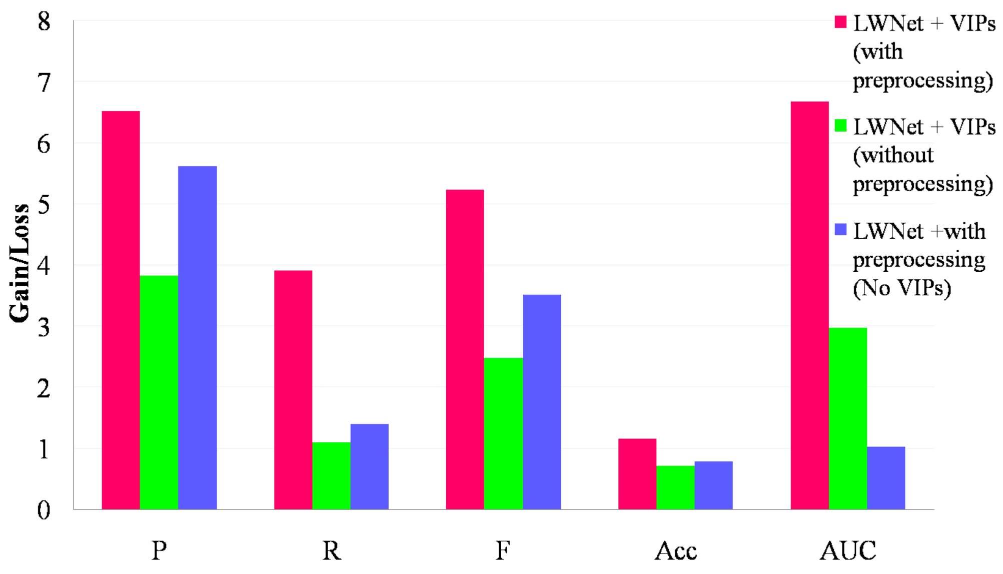

Figure 4.8: LWNet [6] with/without VIPs and with/without preprocessing (CLAHEGC): Graphical representation of the average gains/loss in Table 4.11, in P, R, F, Acc, and AUC for the ablation study.

In the second cross-dataset ablation study, we discuss the hyper-parameters and patches-versus-swirls of our method using two cross-dataset cases: HRF to DRIVE and HRF to CHASEDB. We perform these experiments on LWNet [6] baseline without preprocessing to understand the contribution of our VIPs and their various components towards performance gains. The hyper-parameters are α to control the number of patches, β as minimum intensity level, γ as maximum swirl strength, and the p-value, the augmentation probability at each epoch. We vary one hyper-parameter while keeping the others constant. Based on Table 4.12, VIPs perform best on average with proposed hyper-parameters of α=0.4, β=0.5, γ=10, and p=0.5, that is, they best simulate vessel-like contours.

	HRF [3] to CHASEDB [2]					HRF [3] to DRIVE [1]					Average				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
LWNet [6]	58.54	57.02	57.77	91.32	43.05	71.95	74.77	73.33	95.07	53.19	65.25	65.90	65.55	93.20	48.12
α=variable, β=0.5, γ=10, and p=0.5															
LWNet [6] + VIPs (α=0.1)	57.62	51.33	54.30	90.98	40.23	73.42	73.91	73.66	94.71	54.19	65.52	62.62	63.98	92.85	47.21
LWNet [6] + VIPs (α=0.4, Ours)	61.24	58.76	59.97	93.97	47.38	74.33	75.78	75.05	95.58	63.27	67.78	67.27	67.51	94.77	55.32
LWNet [6] + VIPs (α=0.6)	52.21	52.40	52.30	92.66	41.18	72.19	72.13	72.16	95.12	61.43	62.20	62.26	62.23	93.89	51.30
α=0.4, β=variable, γ=10, and p=0.5															
LWNet [6] + VIPs (β=0.25)	58.61	56.20	57.38	93.59	46.42	70.36	72.21	71.28	94.9	56.54	64.49	64.21	64.33	94.25	51.48
LWNet [6] + VIPs (β=0.75)	50.17	59.98	54.64	92.35	40.98	71.98	75.77	73.83	95.19	63.24	61.08	67.88	64.24	93.77	52.11
LWNet [6] + VIPs (β=1, only swirls)	62.44	53.44	57.59	93.95	47.14	73.06	74.79	73.92	95.37	61.10	67.75	64.12	65.76	94.66	54.12
α=0.4, β=0.5, γ=variable, and p=0.5															
LWNet [6] + VIPs (γ=0, only patches)	57.61	57.09	57.35	93.66	46.27	72.98	74.30	73.64	94.62	54.61	65.30	65.70	65.50	94.14	50.44
LWNet [6] + VIPs (γ=20)	53.94	54.56	54.25	92.93	43.83	72.04	75.66	73.81	95.29	63.33	62.99	65.11	64.03	94.11	53.58
α=0.4, β=0.5, γ=10, and p=variable															
LWNet [6] + VIPs (p=0.25)	60.59	57.83	59.17	93.11	47.01	68.49	77.40	72.67	94.73	57.38	64.54	67.62	65.92	93.92	52.19
LWNet [6] + VIPs (p=0.75)	51.57	52.97	52.26	92.03	40.80	74.01	74.62	74.32	94.34	62.91	62.79	63.79	63.29	93.18	51.85

Table 4.12: Hyper-parameters and ablation (swirls versus patches): of our VIPs with LWNet [6] (no preprocessing). Red indicates best, blue second best, and green third best result respectively.

For α=0.1, false negative predictions increase, indicating that the number of patches is insufficient for the model to distinguish between the contours and thin vessel features. At α=0.6, we observe major losses in P, R, and F compared to the baseline, indicating that too many patches cause loss (masking) of critical features in training data.

At β=1, there is no intensity variation between the patches (i.e., only swirls are induced), and when γ=0, there is no swirl component. In both cases, we observe comparable performance to the model baseline, and this shows that stand alone, neither of them is sufficient for accurate replication of the contours we want our model to learn.

For p=0.25, the performance is comparable to LWNet [6] baseline; this is expected since the original augmentation regime of LWNet [6] generates most training images here, and at p=0.75 we see performance drops of greater than 2% because the model doesn't get enough original images for training.

4.7 Visual Results

Figure 4.9 shows visual results. Upon closer inspection of the magnified patches shown in Figure 4.9, it is evident that our proposed VIPs method is capable of segmenting a significantly larger number of microvessels as compared to the baseline, second [9], and third [10] best related augmentation methods. This supports our claim that using VIPs aids with the segmentation of the real microvessels.

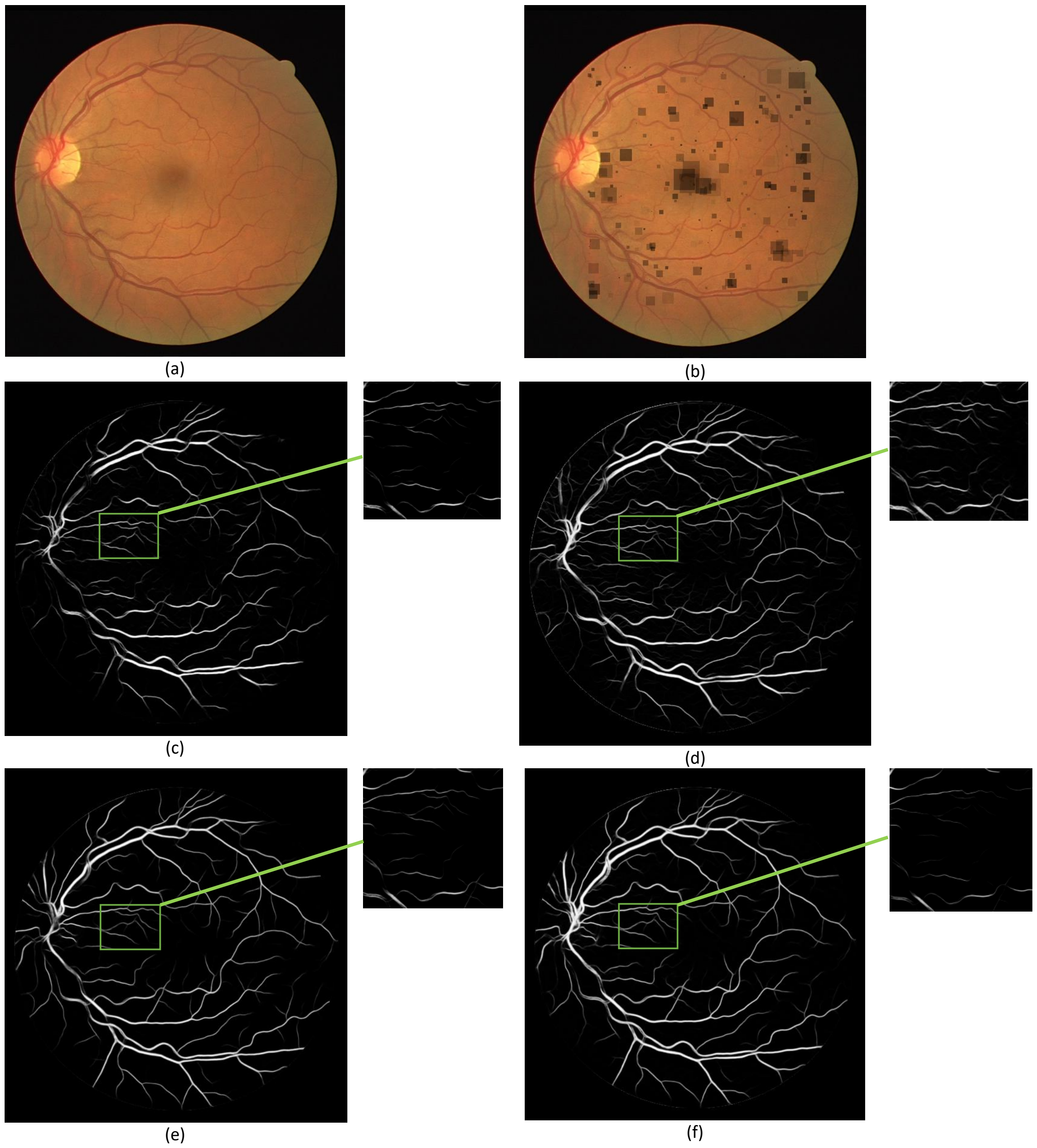

Figure 4.9: (a) Original DRIVE [1] image. (b) Original image in (a) augmented with our VIPs. (c) Segmented original image in (a) using LWNet [6]. (d) Segmented original image in (a) using LWNet [6] trained with VIPs+preprocessing. (e) Segmented original image in (a) using LWNet [6] trained with Robust [9] (second best augmentation method)+preprocessing. (f) Segmented original image in (a) using LWNet [6] trained with Cutmix [10] (third best augmentation method) + preprocessing. Note that the same preprocessing was used for all methods.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Datasets in retinal vessel segmentation (RVS) are small, causing overfitting of RVS models. Data augmentation can reduce overfitting. However, related augmentation methods for RVS do not address the problem of localized variable intensities in Retinal vessel images nor augment patterns to handle microvessels. This thesis proposed a method to reduce overfitting (improve the generalization) of models in RVS through data preprocessing and augmentation. Our data augmentation used intensity-size variant patches with swirls. The swirls simulated the background contours in retinal images to address the class imbalance between real microvessels and background contours. The variable patches were introduced to address pixel misclassification in regions with intensity changes. The thesis also examined the impact of different types of data augmentation and their mode (online versus offline). Furthermore, we studied the effect of preprocessing training samples before augmentation and proposed using contrast enhancement and gamma correction (CLAHEGC) on the grayscale images, as in [7], for preprocessing.

Simulations demonstrated that using the proposed variable intensity patches (VIPs) with preprocessing improved the performance of state of the art RVS models. For example, in terms of AUC, LWNet improved by 6.68% and SegRNet by 5.29%, and in terms of F-score, LWNet improved by 5.21% and SegRNet by 1.03%. Our method is stable (boost performance) across RVS models and also across RVS datasets. Our simulation showed that augmenting with both swirls and patches is more effective than using only one.

Furthermore, our VIPs outperformed other augmentation techniques; for example, compared to the second best method Robust, VIPS improved LWNet by 6.68% versus 5.02% in terms of AUC. We noticed that our method added to the baseline is more stable than related augmentation methods, that is, adding VIPs to different RVS models always achieved gains on average, while adding related augmentation method can produce major losses. We experimentally showed that our VIPs not only helps to reduce the training and validation losses of the RVS models, but also reduces the gap between training and validation losses during training.

5.2 Future Work

Our VIPs augmentation method shown significant improvement for existing RVS models as it enhances their generalization capabilities. Datasets available in the RVS domain are very small. Therefore, there is a large scope of testing our VIPs augmentation method for its efficacy on voluminous RVS datasets that may be available in the future.

While developing our augmentation method, we observed that selecting the hyperparameters for our VIPs methods was a crucial but time-consuming process. We propose exploring techniques of selecting hyperparameters such as AutoAugment [60], grid search, and random grid search [61].

Furthermore, given the overlap between the RVS and crack segmentation domains we propose that, exploring our technique on crack segmentation methods and datasets may prove to be beneficial. There is a future possibility for the applicability of our method, to enhance the performance of curvilinear object segmentation methods like JTFN [52], which claim to be capable of segmenting both cracks and retinal vessels.

Bibliography

[1] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken, “Ridge-based vessel segmentation in color images of the retina,” *IEEE Trans. Med. Imag.*, vol. 23, no. 4, pp. 501–509, 2004.

[2] M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. Rudnicka, C. Owen, and S. Barman, “An Ensemble Classification-Based Approach Applied to Retinal Blood Vessel Segmentation,” *IEEE Trans. Biomed. Eng.*, vol. 59, no. 9, pp. 2538–2548, 2012.

[3] A. Budai, R. Bock, A. Maier, J. Hornegger, and G. Michelson, “Robust Vessel Segmentation in Fundus Images,” *Int. J. Biomed. Imaging*, vol. 2013, p. 154860, Dec 2013.

[4] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *Med Image Comput Comput Assist Interv*, pp. 234–241, Springer International Publishing, 2015.

[5] K. J. Zuiderveld, “Contrast limited adaptive histogram equalization,” in *Graphics gems*, 1994.

[6] A. Galdran, A. Anjos, J. Dolz, *et al.*, “State-of-the-art retinal vessel segmentation with minimalistic models,” *Sci Rep 12*, vol. 6174, 2022.

[7] J. Ryu, M. U. Rehman, I. F. Nizami, and K. T. Chong, "SegR-Net: A deep learning framework with multi-scale feature fusion for robust retinal vessel segmentation," *Comput. Biol. Med.*, vol. 163, p. 107132, 2023.

[8] Li,L. and Verma,M. and Nakashima,Y. and Nagahara,H. and Kawasaki,R., "Iternet: Retinal image segmentation utilizing structural redundancy in vessel networks," in *Proc. IEEE Winter Conf. App. Computer Vision*, pp. 3645–3654, 2020.

[9] X. Sun, H. Fang, Y. Yang, D. Zhu, L. Wang, J. Liu, and Y. Xu, "Robust retinal vessel segmentation from a data augmentation perspective," in *Ophthalmic Medical Image Analysis* (H. Fu, M. K. Garvin, T. MacGillivray, Y. Xu, and Y. Zheng, eds.), (Cham), pp. 189–198, Springer International Publishing, 2021.

[10] S. Yun, D. Han, S. Chun, S. Oh, Y. Yoo, and J. Choe, "CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features," in *Proc. IEEE Int. Conf. Computer Vision*, (Los Alamitos, CA, USA), pp. 6022–6031, nov 2019.

[11] X. Yang, Z. Li, Y. Guo, and D. Zhou, "DCU-net: a deformable convolutional neural network based on cascade U-net for retinal vessel segmentation," *Multimedia Tools and Applications*, vol. 81, pp. 15593–15607, May 2022.

[12] W. Liu, H. Yang, T. Tian, Z. Cao, X. Pan, W. Xu, Y. Jin, and F. Gao, "Full-Resolution Network and Dual-Threshold Iteration for Retinal Vessel and Coronary Angiograph Segmentation," *IEEE J. Biomed. Inform.*, vol. 26, no. 9, pp. 4623–4634, 2022.

[13] F. Dong, D. Wu, C. Guo, S. Zhang, B. Yang, and X. Gong, "CRAUNet: A cascaded residual attention U-Net for retinal vessel segmentation," *Comput. Biol. Med.*, vol. 147, p. 105651, 2022.

[14] O. Sule, “A Survey of Deep Learning for Retinal Blood Vessel Segmentation Methods: Taxonomy, Trends, Challenges and Future Directions,” *IEEE Access*, vol. 10, pp. 38202–38236, 2022.

[15] J. Cervantes, J. Cervantes, F. GarcÃŋa-Lamont, A. Yee-Rendon, J. E. Cabrera, and L. D. Jalili, “A comprehensive survey on segmentation techniques for retinal vessel segmentation,” *Neurocomputing*, vol. 556, p. 126626, 2023.

[16] S. Deari, I. Oksuz, and S. Ulukaya, “Importance of Data Augmentation and Transfer Learning on Retinal Vessel Segmentation,” in *2021 29th Telecommunications Forum (TELFOR)*, pp. 1–4, 2021.

[17] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random Erasing Data Augmentation,” in *Proc. AAAI Conf. Artificial Intelligence*, vol. 34, pp. 13001–13008, Apr. 2020.

[18] F. Dadboud, V. Patel, V. Mehta, M. Bolic, and I. Mantegh, “Single-stage uav detection and classification with yolov5: Mosaic data augmentation and panet,” in *2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–8, 2021.

[19] H. Wang, G. Xu, X. Pan, Z. Liu, N. Tang, R. Lan, and X. Luo, “Attention-inception-based U-Net for retinal vessel segmentation with advanced residual,” *Computers & Electrical Engineering*, vol. 98, p. 107670, 2022.

[20] Y. Liu, J. Shen, L. Yang, H. Yu, and G. Bian, “Wave-Net: A lightweight deep network for retinal vessel segmentation from fundus images,” *Comput. Biol. Med.*, vol. 152, p. 106341, 2023.

[21] X. Li, J. Ding, and J. Tang, “Res2Unet: A multi-scale channel attention network for retinal vessel segmentation.,” *Neural Comput & Applic*, vol. 34, 2022.

[22] T. Shi, N. Boutry, Y. Xu, and T. Geraud, "Local Intensity Order Transformation for Robust Curvilinear Object Segmentation," *IEEE Trans. Image Process.*, vol. 31, pp. 2557–2569, 2022.

[23] A. Bochkovskiy, C. Wang, and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.

[24] G. Ghiasi, T. Lin, and Q. Le, "DropBlock: A regularization method for convolutional networks," in *Adv. Neural Inf. Process. Syst.* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[25] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-Excitation Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, pp. 2011–2023, aug 2020.

[26] S. Guo, "CSGNet: Cascade semantic guided net for retinal vessel segmentation," *Biomed. Signal Process. Control*, vol. 78, p. 103930, 2022.

[27] Z. Zhu, Q. An, Z. Wang, Q. Li, H. Fang, and Z. Huang, "ILU-Net: Inception-Like U-Net for retinal vessel segmentation," *Optik*, vol. 260, p. 169012, 2022.

[28] Z. Huang, X. Li, L. Wang, H. Fang, L. Ma, Y. Shi, and H. Hong, "Spatially adaptive multi-scale image enhancement based on nonsubsampled contourlet transform," *Infrared Physics & Technology*, vol. 121, p. 104014, 2022.

[29] T. Jiao, H. Xiang, C. Liu, and X. Zhang, "Algorithm for Retinal Vessel Segmentation based on Axial Transformer and Convolutional Neural Network," in *2022 China Automation Congress (CAC)*, pp. 1462–1466, 2022.

[30] J. Li, G. Gao, L. Yang, and Y. Liu, “GDF-Net: A multi-task symmetrical network for retinal vessel segmentation,” *Biomed. Signal Process. Control*, vol. 81, p. 104426, 2023.

[31] J. Li, G. Gao, L. Yang, G. Bian, and Y. Liu, “DPF-Net: A Dual-Path Progressive Fusion Network for Retinal Vessel Segmentation,” *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–17, 2023.

[32] A. G. Roy, N. Navab, and C. Wachinger, “Recalibrating Fully Convolutional Networks with Spatial and Channel ’Squeeze & Excitation’ Blocks,” *CoRR*, vol. abs/1808.08127, 2018.

[33] J. Li, G. Gao, Y. Liu, and L. Yang, “MAGF-Net: A multiscale attention-guided fusion network for retinal vessel segmentation,” *Measurement*, vol. 206, p. 112316, 2023.

[34] Y. Li, Y. Zhang, J.-Y. Liu, K. Wang, K. Zhang, G.-S. Zhang, X.-F. Liao, and G. Yang, “Global Transformer and Dual Local Attention Network via Deep-Shallow Hierarchical Feature Fusion for Retinal Vessel Segmentation,” *IEEE Transactions on Cybernetics*, vol. 53, no. 9, pp. 5826–5839, 2023.

[35] Y. Liu, J. Shen, L. Yang, G. Bian, and H. Yu, “ResDO-UNet: A deep residual network for accurate retinal vessel segmentation from fundus images,” *Biomed. Signal Process. Control*, vol. 79, p. 104087, 2023.

[36] D. Alvarado-Carrillo and O. Dalmau-Cedeno, “Width Attention based Convolutional Neural Network for Retinal Vessel Segmentation,” *Expert Syst. Appl.*, vol. 209, p. 118313, 2022.

[37] A. Desiani, Erwin, B. Suprihatin, F. Efriliyanti, M. Arhami, and E. Setyaningsih, “VG-DropDNet a Robust Architecture for Blood Vessels Segmentation on Retinal Image,” *IEEE Access*, vol. 10, pp. 92067–92083, 2022.

[38] G. Huang, Z. Liu, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” *CoRR*, vol. abs/1608.06993, 2016.

[39] M. Arsalan, T. M. Khan, S. S. Naqvi, M. Nawaz, and I. Razzak, “Prompt Deep Light-Weight Vessel Segmentation Network (PLVS-Net),” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 20, no. 2, pp. 1363–1371, 2023.

[40] S. Deari, I. Oksuz, and S. Ulukaya, “Block Attention and Switchable Normalization Based Deep Learning Framework for Segmentation of Retinal Vessels,” *IEEE Access*, vol. 11, pp. 38263–38274, 2023.

[41] P. Luo, R. Zhang, J. Ren, Z. Peng, and J. Li, “Switchable Normalization for Learning-to-Normalize Deep Representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, pp. 712–728, feb 2021.

[42] Y. Ye, C. Pan, Y. Wu, S. Wang, and Y. Xia, “MFI-Net: Multiscale Feature Interaction Network for Retinal Vessel Segmentation,” *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 9, pp. 4551–4562, 2022.

[43] J. Hu, L. Shen, and G. Sun, “Squeeze-and-Excitation Networks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

[44] S. Guo, K. Wang, H. Kang, Y. Zhang, Y. Gao, and T. Li, “BTS-DSN: Deeply supervised neural network with short connections for retinal vessel segmentation,” *Int. J. Med. Inform.*, vol. 126, pp. 105–113, 2019.

[45] V. Cherukuri, V. Kumar B.G., R. Bala, and V. Monga, “Deep Retinal Image Segmentation With Regularization Under Geometric Priors,” *IEEE Trans. Image Process.*, vol. 29, pp. 2552–2567, 2020.

[46] S. Gao, M. Cheng, K. Zhao, X. Zhang, M. Yang, and P. Torr, “Res2Net: A New Multi-Scale Backbone Architecture,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 43, pp. 652–662, feb 2021.

[47] L. Mou, L. Zhao, Y.and Chen, J. Cheng, Z. Gu, H. Hao, H. Qi, Y. Zheng, A. Frangi, and J. Liu, “CS-Net: channel and spatial attention network for curvilinear structure segmentation,” in *Int. Conf. on Med. Imag. Computing and Computer-Assisted Intervention*, pp. 721–730, Springer, 2019.

[48] D. Alvarado-Carrillo, E. Ovalle-Magallanes, and O. Dalmau-Cedeño, “D-GaussianNet: Adaptive Distorted Gaussian Matched Filter with Convolutional Neural Network for Retinal Vessel Segmentation,” in *Geometry and Vision* (M. Nguyen, W. Q. Yan, and H. Ho, eds.), (Cham), pp. 378–392, Springer International Publishing, 2021.

[49] Y. Zhou, Z. Chen, H. Shen, X. Zheng, R. Zhao, and X. Duan, “A refined equilibrium generative adversarial network for retinal vessel segmentation,” *Neurocomputing*, vol. 437, pp. 118–130, 2021.

[50] C. Rajesh, R. Sadam, and S. Kumar, “An evolutionary U-shaped network for Retinal Vessel Segmentation using Binary Teaching-Learning-Based Optimization,” *Biomed. Signal Process. Control*, vol. 83, p. 104669, 2023.

[51] A. Taheri, K. RahimiZadeh, and R. Rao, “An efficient Balanced Teaching-Learning-Based optimization algorithm with Individual restarting strategy for solving global optimization problems,” *Inf. Sci.*, vol. 576, pp. 68–104, 2021.

[52] M. Cheng, K. Zhao, X. Guo, Y. Xu, and J. Guo, “Joint topology-preserving and feature-refinement network for curvilinear structure segmentation,” in *Proc. IEEE Int. Conf. Computer Vision*, pp. 7127–7136, 2021.

[53] S. van der Walt, J. Schonberger, J. Nunez-Iglesias, F. Boulogne, J. Warner, N. Yager, E. Gouillart, and T. Yu, “scikit-image: image processing in Python.,” *PeerJ 2:e453*, 2014.

[54] Y. Yan, S. Zhu, S. Ma, Y. Guo, and Z. Yu, “CycleADC-Net: A crack segmentation method based on multi-scale feature fusion,” *Measurement*, vol. 204, p. 112107, 2022.

[55] Z. Qu, L. Zhuo, J. Cao, X. Li, H. Yin, and Z. Wang, “TP-Net: Two-Path Network for Retinal Vessel Segmentation,” *IEEE J. Biomed. Inform.*, vol. 27, no. 4, pp. 1979–1990, 2023.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[57] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[58] F. Milletari, N. Navab, and S.-A. Ahmadi, “V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation,” in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 565–571, 2016.

[59] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 39, no. 12, pp. 2481–2495, 2017.

[60] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation strategies from data,” in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2019.

[61] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012.

Appendix A

Additional In-Dataset Comparison

Based on the first half of the in-dataset average results in Table A.1 we observe that LWNet [6] is only slightly better than the larger SegRNet [7] in terms of F-score by 0.21% and AUC by 0.36%. The average values of the second half of Table A.1 show that our proposed VIPs improves LWNet [6], SegRNet [7], and IterNet [8] by 0.45%, 0.64%, and 0.31% respectively in terms of F-score, also we see gains of 0.13%, 0.54%, and 0.34% in terms of AUC.

	DRIVE [1]					HRF [3]					CHASEDB [2]				
Method	P	R	F	Acc	AUC	P	R	F	Acc	AUC	P	R	F	Acc	AUC
Baselline															
IterNet [8]	81	81.21	81.08	96.52	61.35	79.28	78.58	78.75	94.01	61.02	79.51	82.01	80.1	96.21	63.2
SegRNet [7]	82.06	83.46	82.27	96.91	64.08	80.06	80.4	80.23	95.96	65.35	82.41	85.29	83.83	97.81	64.04
LWNet [6]	81.50	82.79	82.14	97.05	63.81	79.02	80.84	79.92	96.74	64.22	85.50	86.16	85.83	97.78	66.52
Baselline + VIPs															
IterNet [8] + VIPs	80.77	81.9	80.92	96.61	61.65	79.46	78.39	78.92	94.09	60.99	80.01	82.1	81.04	96.77	63.96
	(-0.23)	(0.69)	-(0.16)	(0.09)	(0.3)	(0.18)	(-0.19)	(0.17)	(0.08)	-(0.03)	(0.5)	(0.09)	(0.94)	(0.56)	(0.76)
SegRNet [7] + VIPs	81.87	83.83	82.84	96.95	64.83	80.37	81.81	81.05	95.99	65.55	82.91	85.86	84.35	97.95	64.72
	(-0.19)	(0.37)	(0.57)	(0.04)	(0.75)	(0.31)	(1.41)	(0.82)	(0.03)	(0.2)	(0.5)	(0.57)	(0.52)	(0.14)	(0.68)
LWNet [6] + VIPs	81.76	83.68	82.71	97.08	64.01	79.63	81.28	80.44	97.03	64.81	85.54	86.67	86.10	97.85	66.63
	(0.26)	(0.89)	(0.57)	(0.03)	(0.2)	(0.61)	(0.44)	(0.52)	(0.29)	(0.59)	(0.04)	(0.51)	(0.271)	(0.07)	(0.11)

	Average				
Method	P	R	F	Acc	AUC
Baselline					
IterNet [8]	79.93	80.60	79.98	95.58	61.86
SegRNet [7]	81.51	83.05	82.11	96.89	64.49
LWNet [6]	82.01	83.26	82.63	97.19	64.85
Baselline + VIPs					
IterNet [8] + VIPs	80.08	80.80	80.29	95.82	62.20
	(0.15)	(0.2)	(0.31)	(0.24)	(0.34)
SegRNet [7] + VIPs	81.72	83.83	82.75	96.96	65.03
	(0.21)	(0.78)	(0.64)	(0.07)	(0.54)
LWNet [6] + VIPs	82.31	83.88	83.08	97.32	65.15
	(0.3)	(0.62)	(0.45)	(0.13)	(0.3)

Table A.1: Results for in-dateset analysis of LWNet [6], SegRNet [7], and IterNet [8] both with and without VIPs. We give the gains/loss over baseline in (). (We used CLAHEGC as preproessing.)